VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

SBĚR MAR DAT
COLLECTION OF MEASUREMENT AND CONTROL DATA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Matouš Rathouzský**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Radek Štohl, Ph.D.**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Matouš Rathouzský ***ID:*** 165344
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Sběr MaR dat

POKYNY PRO VYPRACOVÁNÍ:

1. Proveďte literární rešerši o sběru dat z inteligentních budov.
2. Seznamte se se softwarem COACH AX.
3. Realizujte databázový model pro ukládání MaR dat z budovy T12 podle technologických schémat.
4. Realizujte generování simulačních dat.
5. Vytvořte příslušnou vizualizaci MaR dat.
6. Ověřte své řešení.

DOPORUČENÁ LITERATURA:

VALEŠ, M., Inteligentní dům, 2. vyd., ERA, 2008, ISBN: 978-80-7366-137-3
BACnetTM – A standard communication infrastructure for intelligent buildings [online]. Dostupné z: http://www.bacnet.org/Bibliography/AIC-97/AIC1997.htm

Dle vlastního literárního průzkumu a doporučení vedoucího práce.

Termín zadání: 5. 2. 2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: Ing. Radek Štohl, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta elektrotechniky a komunikačních technologií
Ústav automatizace a měřicí techniky

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Abstrakt

Tato bakalářská práce nastiňuje představu o pojmu inteligentní budova a s ní související sběr MaR dat. V první části je popsán komunikační protokol BACnet a sbíraná data z inteligentních budov. Dále je popsán software COACH^{AX} od společnosti Honeywell, který je nedílnou součástí praktické části bakalářské práce. V poslední části je realizace databázového modelu pro ukládání simulovaných MaR dat budovy T12 Fakulty elektrotechniky a komunikačních technologií a následně je vytvořeno generování simulovaných dat a příslušná vizualizace.

Klíčová slova

Inteligentní budova, databáze, COACH^{AX}, MS-SQL Server, generování dat

Abstract

This bachelor thesis outlines the concept of intelligent buildings and the associated collection of measurement and control data. The first part describes the BACnet communication protocol and data collected from the intelligent building. It also describes Honeywell's COACH^{AX} software, which is an integral part of the practical part of the bachelor thesis. The last part is the implementation of a database model for the storage of simulated measurement and control data T12 at the Faculty of Electrical Engineering and Communication Technologies and afterwards is generate simulated data and created visualization.

Keywords

Intelligent building, database, COACH^{AX}, MS-SQL Server, data generation

Bibliografická citace:

RATHOUZSKÝ, M. *Sběr MaR dat*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2018. 46s. Vedoucí bakalářské práce Ing. Radek Štohl, Ph.D..

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Sběr MaR dat jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **21. května 2018**

……
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Radku Štohlovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: **21. května 2018**

……
podpis autora

Obsah

Seznam obrázků

ÚVOD

Za posledních sto let jsme nesmírně pokročili v technice a stala se nedílnou součástí našich životů. Nejvíce se to projevilo v budovách, kde člověk tráví více jak 70 % svého času. Zde se můžeme setkat s různými inteligentními systémy, které mají za cíl zvýšit komfort jejich uživatelům a zefektivnit energetický provoz, většinou jsou označovány zkratkou MaR a přesněji řečeno slouží k řízení a optimalizaci zařízení sloužících na vytápění, klimatizování či řízení různých technologických procesů. V budově T12 Fakulty elektrotechniky a komunikačních technologií tomu není jinak.

Cílem této práce je seznámit čtenáře jakým způsobem lze sbírat data z inteligentních budov, realizace databázového modelu ukládání MaR dat z budovy T12 podle technologických schémat, realizace generování dat a tvorba příslušné vizualizace.

1. INTELIGENTNÍ BUDOVA

Aniž bychom si to uvědomovali, s inteligentními budovami se setkáváme dnes už prakticky denně. Existuje mnoho různých definic, které popisují, co je inteligentní budova, ale prakticky tyto definice jde shrnout následně.

Inteligentní budova [1] je objekt vybavený počítačovou a komunikační technikou, která předvídá a reaguje na potřeby obyvatel s cílem zvýšit jejich komfort, pohodlí, snížit spotřebu energií, poskytnout jim bezpečí a zábavu pomocí řízení všech technologií v domě a jejich interakcí s vnějším světem.

V následující práci budeme pracovat s daty budovy Elektrotechnické fakulty areálu Technické 12, a proto bych si dovolil napsat, že se jedná o specifický typ budovy, jakožto administrativní, obsahující nejméně z 50 % svého obestavěného prostoru provozy kancelářského charakteru, určené pro studijní činnost. Nejvíce je tu tedy kladen důraz na komfort studentů, vyplívající např. se zajištěním čerstvého přísunu vzduchu, kterou zajišťují vzduchotechnické jednotky, známe také pod anglickou zkratkou jako AHU (air handling unit).

Z pohledu řízení budovy [2] se jedná o jeden komplexní celek, ve kterém jsou systémy propojeny do jedné komunikační/vizualizační platformy a řízené technologie jsou schopny spolu vzájemně komunikovat. Srdcem budovy je zcela jednoznačně sytém nazývaný MaR (provozní soubor měření a regulace) integrovaný do řídicího systému budovy (Building Management System, BMS).

1.1 Komunikace technologií

Na rozdíl od komunikace mezi lidmi prostřednictvím jazyka se v technické komunikaci jedná o přenos dat mezi přístroji automatizační techniky za pomocí sběrnic [6] a komunikačních protokolů [6].

Protokol *„Soubor pravidel pro komunikaci mezi dvěma nebo více uzly (systémy, regulátory).“*

Sběrnice *„(anglicky: Bus) je skupina signálových vodičů, kterou lze rozdělit na skupiny řídicích, adresových a datových vodičů v případě paralelní sběrnice nebo sdílení dat a řízení na společném vodiči (nebo vodičích) u sériových sběrnic. Sběrnice má za účel zajistit přenos dat a řídicích povelů mezi dvěma a více elektronickými zařízeními. Přenos dat na sběrnici se řídí stanoveným protokolem. „*

Pro automatizaci a řízení budov se zejména používají sběrnicové systémy KNX/EIB, LON a dále komunikační protokoly typu BACnet a Modbus. V následující podkapitole si více přiblížíme protokol BACnet, který je hojně využíván v budově areálu Technické 12.

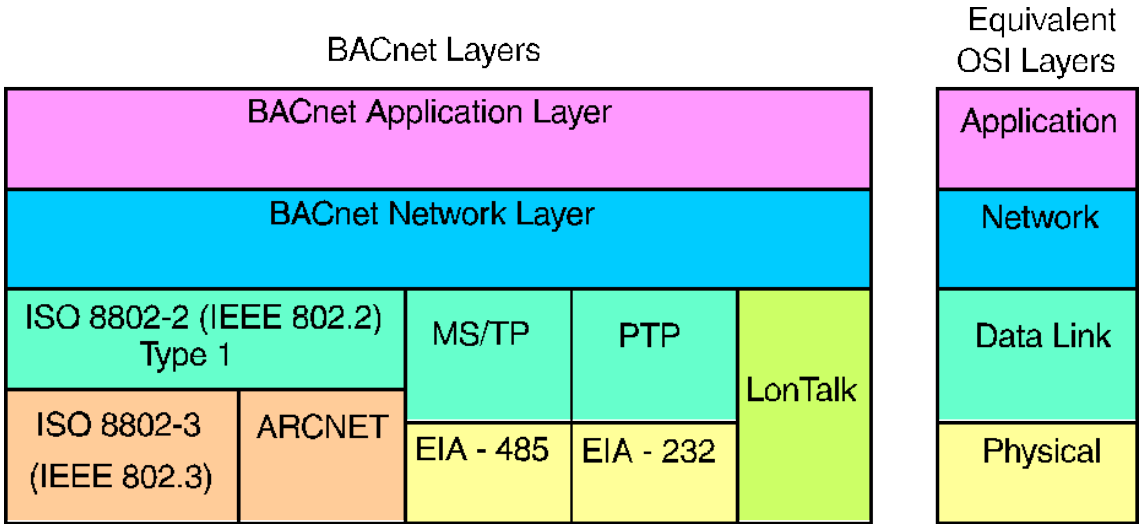
1.1.1 Protokol BACnet

BACnet (Building Automation and Controls Network) je standardní komunikační protokol pro sítě automatizace a řízení budov, vyvinutý americkým sdružením ASHRAE (American Society of Heating, Refrigerating and Air-conditioning Engineers). Protokol BACnet se používá pro řízení systému topení, ventilace, klimatizace, řízení osvětlení a pro různé bezpečnostní systémy, jako například pro odhalení požáru, alarmy a jiné systémy.

Z [7] cituji: „*Protokol BACnet stanovuje standardní způsoby, jak reprezentovat funkce (data) libovolného zařízení jako například analogové a binární vstupy a výstupy, časové programy, řídící smyčky a alarmy. BACnet nedefinuje interní konfiguraci, datové struktury nebo řídící logiku zařízení. Informace poskytované na síti BACnet jsou definovány jako standardizované abstraktní objekty. Vazba těchto objektů na reálně naměřené hodnoty je definována výrobcem. Stejné pravidlo platí i pro implementaci řídících algoritmů zařízení, standardizováno je rozhraní vzhledem k síti BACnet, vnitřní architektura není pro standard BACnet podstatná.*“

Architektura BACnet a model ISO/OSI

Specifikace BACnet je složena v zásadě ze tří hlavních částí. První část popisuje metody, jak reprezentovat jakékoli zařízení standardním způsobem (tj. *objekty*). Druhá část definuje zprávy zasílané počítačovou sítí pro monitoring a řízení takového zařízení (tj. *služby*). Třetí část definuje množinu přístupných lokálních sítí (LAN) použitelných pro přenos zpráv. Vlastní architektura BACnetu je založena na modelu ISO/OSI (Obrázek 1.1). BACnet rovněž umožňuje směrovat zprávy skrze existující IP sítě a Novell IPX sítě. Oba tyto protokoly jsou schopny zapouzdřit BACnet zprávy a přenést je pomocí tzv. tunelování (BACnet/IP Packet-Assembler-Disassembler: B/IP PAD).



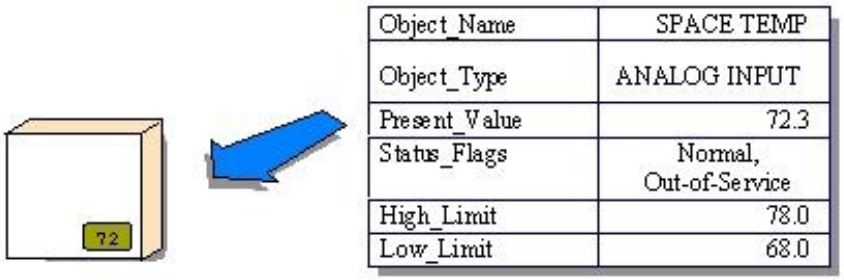
Obrázek 1.1: Architektura komunikačního protokolu BACnet [3]

BACnet objekty

Jednotlivá zařízení, přístroje, jednotky a snímače jsou v síti BACnet vždy reprezentovány jedním či skupinou tzv. BACnet objektů [4]. Každý je pak charakterizován nastavením vlastností, které popisují jeho chování a řídí jejich provoz.

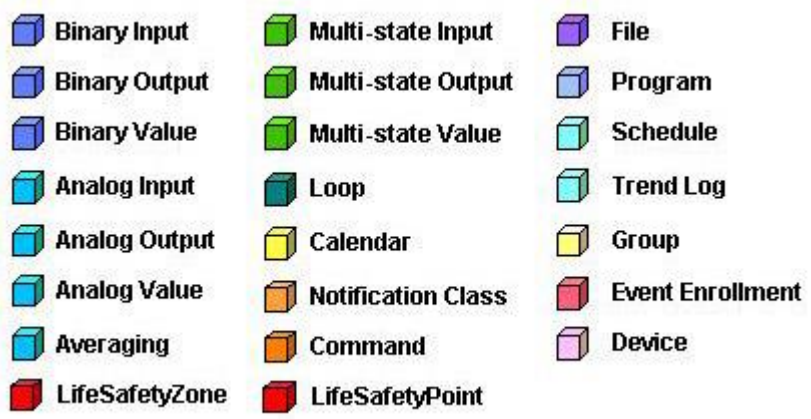
Všechny objekty BACnetu poskytují sadu vlastností, které se využívají pro organizaci, poskytování a získávání informací o daném objektu a řízení objektu. Objekt si můžete představit jako tabulku se dvěma sloupci, kde nalevo jsou jména jednotlivých informačních položek a napravo údaj jejich o vlastnosti či hodnotě. Zatímco některé položky jsou určeny pouze pro čtení, lze se na ně podívat, ale nelze je měnit, tak některé umožňují zápis / změnu stavu či hodnoty.

Můžeme si např. představit objekt reprezentující teplotní čidlo (Obrázek 1.2). Jde o objekt typu analogový vstup („BACnet Analog Input Objekt“). Objekt se jmenuje „Space TEMP“ a tedy reprezentuje typ „ANALOG OBJEKT“. Položka „Present_Value“ obsahuje informaci o aktuální teplotě v °C. Další položky mohou obsahovat další dodatečné informace, jako například o provozním stavu, překročení či podtečení zadaného limitu pro potřeby alarmu apod.



Obrázek 1.2: Ukázka vzhledu tabulky objektu u BACnetu [4]

BACnet definuje kolekci 23 různých standardních typů objektů:



Obrázek 1.3: Standardní objektové typy BACnet [4]

BACnet služby

Jde o definované „příkazy“ umožňující práci s objekty ve smyslu jejich obsluhy, přesněji řečeno definice požadavků informací a úkonů, které může požadovat BACnet jednotka komunikující s jinou BACnet jednotkou. Prakticky tedy jde o příkazy, které ve formě zpráv vysílá jedno BACnet zařízení druhému a tím mu definuje, co od něj požaduje a co má udělat. Protože BACnet je založen na komunikačním modelu „Klient-Server “, jsou zprávy zde nazývány službami (services), které jsou vysílány serverem k jemu příslušným klientům.

Jednotlivé „aplikační služby“ jsou pro přehlednost podlé své funkce rozděleny do několika skupin (tříd):

- Služby hlášení (Alarmů) a událostí (Events)
- Služby přístupu k souborům (File Access Services)
- Služby přístupu k objektům (Object Access Services)
- Služby vzdálené správy zařízení (Remote Device Management Services)
- Služby virtuálního terminálu

BACnet komunikace

Poslední a důležitou částí každé komunikační technologie je samotný systém vzájemného předávání zpráv mezi jednotlivými objekty. V tomto směru BACnet ve většině případů využívá již existujících komunikačních / přenosových protokolů. Jejich sada specifikovaná v BACnetu byla zvolena z důvodu toho, že splňuje celosvětové požadavky řízení a automatizace budov ve smyslu rychlosti, propustnosti, ceny,

uživatelského přístupu atd. Jsou jimi hlavně Ethernet, ARCNET a LongTalk, což jsou zcela soběstačné LAN systémy, vyžadující jen minimální práci se specifikací, jak mají být BACnet zprávy přenášeny.

1.2 Sběr dat

Staré úsloví praví, že měřit znamená vědět. Naměřené hodnoty, data, se stávají podkladem pro další zpracování, jak běžnými statistickými nástroji pro případy energetické optimalizace, tak pokročilými metodami, jako jsou například algoritmy pro detekci poruch.

Systém MaR [2] často zprostředkovává vazby a výměny dat s ostatními systémy nebo technologiemi budov jako zabezpečovací systém, přístupový systém, kamerový systém, systém řízení osvětlení, ovládání rolet a žaluzií, systém záložního napájení důležitých technologií, řízení výtahů, regulace motorů frekvenčními měniči, obnovitelné zdroje energie, rekuperační jednotky a rozvody elektrické energie v celé budově. Veškeré důležité informace z těchto systémů jsou následně ukládány na jednom místě (v řídicí centrále) a jsou zpracovávány do jednodušších výstupů nebo manažerských přehledů různých hodnot a stavů. Řídicí systémy inteligentních budov na některé změny stavu reagují automaticky a ty kvalitnější z nich mají automatické nebo naprogramované optimalizační SW funkce. Po vyhodnocení takto získaných dat můžeme zvolit vhodný režim provozu budovy podle konkrétních potřeb a požadavků s následnou optimalizací spotřeby.

1.2.1 Skupiny měřených dat

Měřené hodnoty a snímané signály můžeme rozdělit do dvou skupin:

Přímé signály poruch

Jedná se většinou o binární vstupy, které přímo hlásí poruchy technologie, například signál z protinámrazového termostatu, diferenčního manostatu na filtru, pomocného kontaktu motorové ochrany atd. Aktivní signál znamená poruchu. Většina těchto signálů je již dostupná v systému MaR, zaintegrovaném do řídicího systému budovy BMS – a data jsou tak k dispozici v databázích historických hodnot. Pokud příslušné signály nejsou vzorkovány, nebývá problém je k trendovaným hodnotám přidat. K přímým signálům poruch také patří vybočení mimo pevné meze analogových hodnot, například pokles tlaku freonu nebo vody v topném či chladicím systému. Tyto signály mohou vznikat již v BMS, protože podmínky pro jejich aktivaci jsou pevně dané (například jako zámraz na zpátečce ohřevu je vyhodnocován pokles teploty v potrubí pod 15 °C).

Odvozené signály poruch

Jsou to výsledky vyhodnocování řady naměřených i historických dat pokročilými statistickými metodami detekce poruch. Podstatné je těmto algoritmům poskytnout středně – až dlouhodobé soubory relevantních dat, což z hlediska měření a regulace představuje okamžité měřené i kumulované hodnoty. Odvozené signály [8] nemají v systému měření a regulace žádný obraz, jsou to hodnoty vypočtené ve statistické nadstavbě nad BMS. V blízké budoucnosti nás, ale díky zvyšujícímu se výpočetnímu výkonu a rostoucí záznamové kapacitě regulačních podstanic čeká prorůstání těchto funkcí i na automatizační úroveň, což představuje významnou změnu v pohledu na detekci poruch vůbec.

1.3 **Databáze**

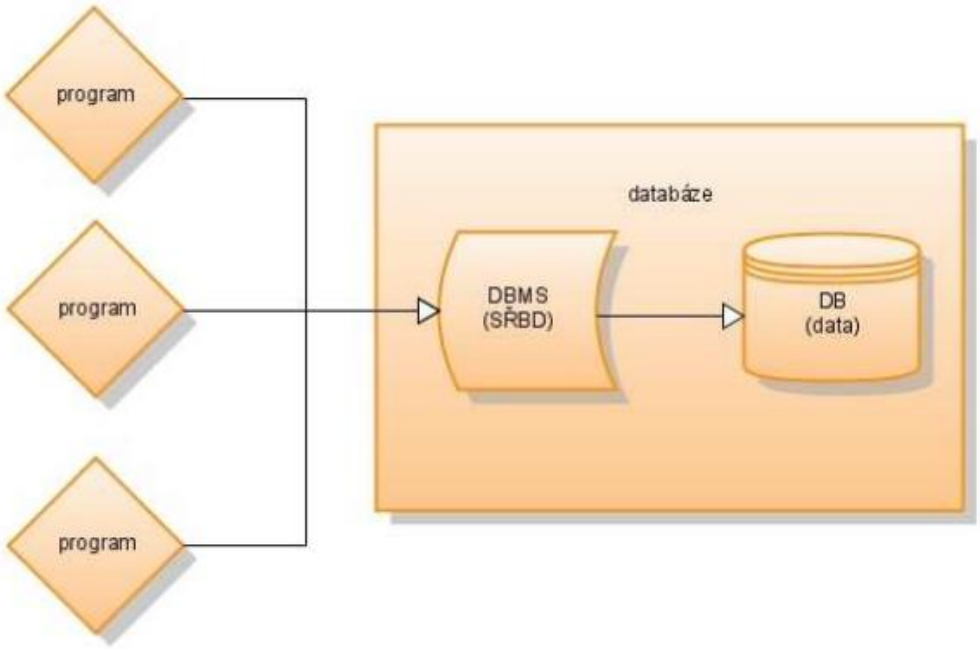
Databázi si lze jednoduše představit jako místo, kam ukládáme získaná data. Mezi hlavní důvody, proč databáze používáme patří např.:

- umožňuje přímý přístup k datům
- má zabudovaný mechanismus pro paralelní přístup k datům
- má zabudovaný systém uživatelských práv
- umožňuje pomocí dotazů snadno extrahovat množiny dat, která vyhovují zadaným kritériím

Velmi jednoduchým příkladem může být knihovna nebo kartotéka, kde jsou data uložena podle určitého systému. Pokud víme, jak tento systém použít, otevírají se nám možnosti, jak efektivně vyhledat data, která zrovna požadujeme.

Databáze a její databázový systém [9] slouží tedy k definici dat, která mají být ukládána. Řeší vztahy mezi těmito daty, způsob, jak je k nim přistupováno a jaké operace je s daty možné realizovat. V neposlední řadě pak funguje jako systém pro řízení přístupu k informacím – řeší oprávnění pro manipulaci s daty a správu uživatelů, jež mají k datům přístup.

Pokud se hovoří o databázích, tak rozlišujeme pojmy DataBase (DB, báze dat) – samotná data a pojem DataBase Management System (DBMS, česky systém řízení báze dat – SŘBD), který se stará o jejich fyzické uložení, správu a požadované operace.



Obrázek 1.4: Databáze pojmy [5]

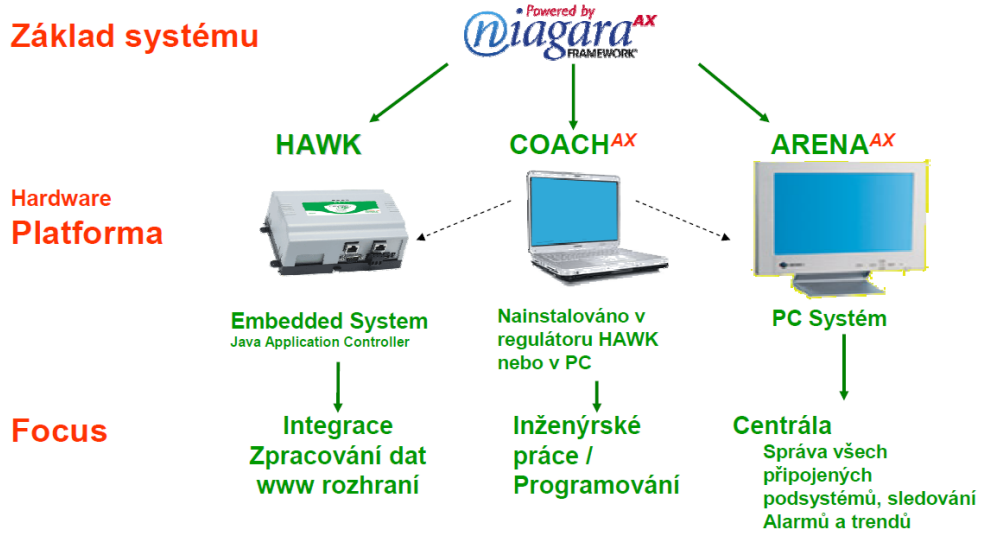
1.3.1 Databázové modely

Ze způsobu ukládání dat a vazeb mezi nimi můžeme rozdělit databáze do základních typů:

- Hierarchická databáze
- Síťová databáze
- Relační databáze
- Objektová databáze
- Objektově relační databáze

2. SOFTWARE COACH^{AX}

COACH^{AX} je multifunkční softwarový vývojový nástroj používaný k programování a ztvárnění řídicích stanic, založený na Niagara frameworku, vyvíjený divizí Centraline, spadající pod firmu Honeywell.



Obrázek 2.1: Řídicí systém Honeywell – CentraLine [6]

Řešení řídicího systému Centrilene [10] si můžeme rozdělit na grafickou centrálu ARENA^{AX} a na regulátory HAWK. Software COACH^{AX} pak primárně slouží k jejich naprogramování. V této kapitole si software více přiblížíme.

2.1 Platforma a stanice

Software COACH^{AX} pro svoji funkci využívá platformu a stanici. Platformou [11] rozumíme jako hlavního zprostředkovatele řízení. Může se buď jednat o PC, na kterém je COACH^{AX} nainstalovaný, nebo HAWK regulátor. Hlavní úlohou platformy je obstarání služeb zapojeným stanicím, které využívají danou platformu. Jinými slovy zvolená platforma rozhoduje, jaké funkce lze u připojených stanic použít. Připojení do platformy je závislé na typu platformy. Pokud se chceme připojit do regulátoru HAWK, používáme jako základní přihlašovací jméno: tridium a heslo: niagara. V případě, že chceme zapojit jako platformu PC, tak se používají administrátorské přihlašovací údaje do operačního systému, který běží v počítači.

K platformě se připojují stanice. Stanice obsahuje aplikace, které se vytváří v grafickém prostředí. Pokud se stanice úspěšně připojí na platformu, aplikace, kterou stanice obsahuje, se aktivuje. V COACH^{AX} lze navrhnout mnoho stanic, ale aktivní může být jen jedna. Pro platformu HAWK lze nahrát a aktivovat pouze jednu stanici.

Základní přihlašování údaje ke stanici jsou pro obě platformy stejné přihlašovací jméno: admin a heslo: volitelné, s tím že musí obsahovat minimálně 10 písmen, alespoň jednu číslici a velké písmeno. Samozřejmě je možné vytvářet vlastní přihlašovací údaje za pomoci Category service [12].

2.2 Služby a aplikace používané v COACH^{AX}

V program COACH^{AX} existuje obrovské množství použitelných aplikací a funkcí, od jednoduchého řízení po aplikace se složitými řídicími algoritmy a náročnou logikou. Systém COACH^{AX} je psaný v programovacím jazyce JAVA a je zde možné doprogramovat své další rozšíření.

2.3 Datové typy

Systém COACH^{AX} podporuje čtyři datové typy:

- Boolean – bitové hodnoty, reprezentovaný zelenou barvou
- Numeric – číselný tvar, reprezentovaný fialovou barvou
- Enumerated – více parametrový typ, reprezentovaný oranžovou barvou
- String – textový charakter, reprezentovaný bílou barvou

Datové typy lze jednoduše přetypovat za pomocí specifických konvertorů. Datové typy mohou být buď ve tvaru konstant nebo přepisovatelných tvarů.

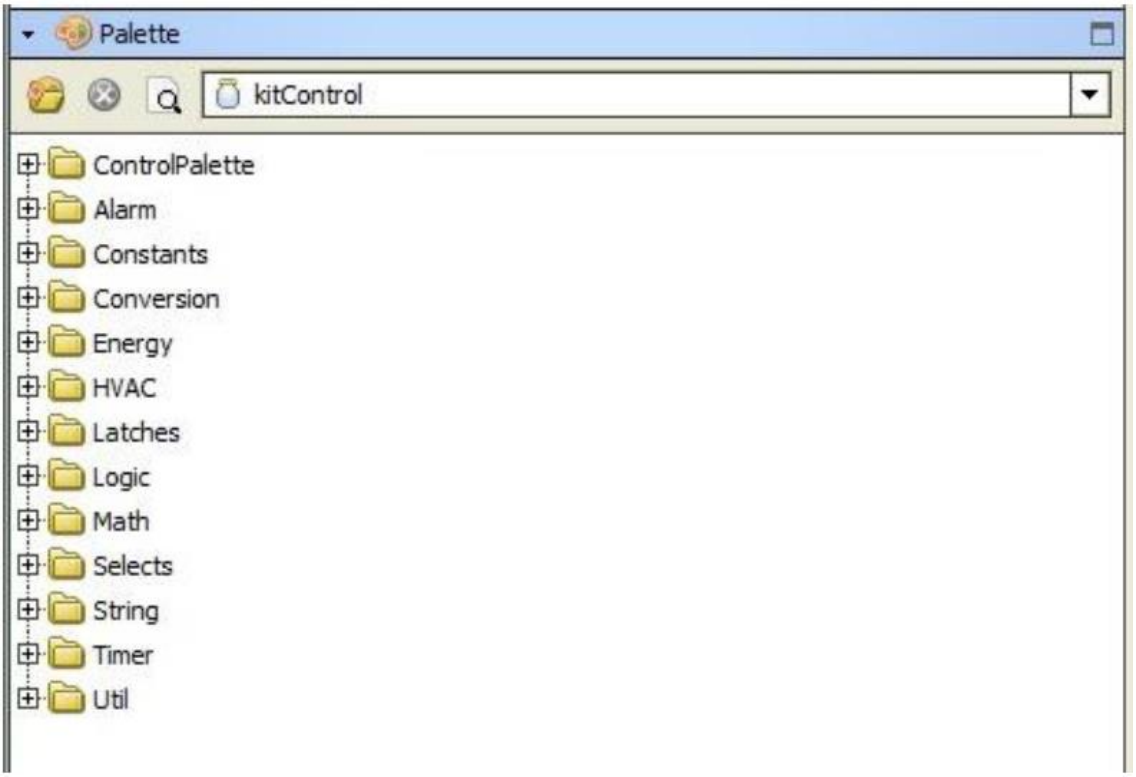
2.4 Knihovny

V prostředí, pod tzv. *Palette*, vidíme knihovny. Zde jsou k dispozici různé předpřipravené prvky [13], jako například prvky grafické pro vzduchotechnické zařízení a různé prvky od identifikačních žárovek po potrubí. Tyto knihovny je možné updatovat, instalovat z neoficiálních zdrojů a mazat v nastavení platformy, konkrétně větvi Software Manager. Abychom mohli využívat grafické prvky, je nutné do palety přidat knihovnu *kitControl*. Prostředí musí být nastaveno na *Graphic* pohled.

V softwaru COACH^{AX} lze použít tyto prvky:

- Matematické funkce – sčítaní, násobení, logaritmy, exponenciála atd.
- Logické funkce – and, or, porovnání, xor atd.
- Časovače
- Generátory – impulzní, sinusový průběh, náhodný generátor čísel a rampa
- Prvky pro řízení HVAC – zpětnovazební smyčka, PID regulátor, hystereze atd.
- Plánovače času

Všechny prvky se dají vybrat za pomocí palety nástrojů. Stačí si hledaný prvek najít v paletě a přetáhnout na vývojovou plochu.



Obrázek 2.2: Ukázka knihovny kitControl [12]

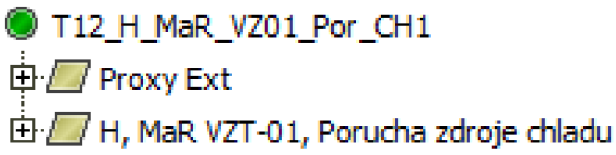
2.5 Alarmy

Alarmy lze připojit k sledovanému prvku. Alarm lze nastavit, aby se aktivoval při překročení povoleného rozsahu. Každý alarm se musí přiřadit ke své třídě. Třídy se vytváří v service/ alarm service. Pro zobrazení alarmu se používá konzolový příjemce, který zobrazuje a ukládá vyskytlé alarmy. Je možné nastavit, aby se vyskytlé alarmy posílaly za pomocí emailu.

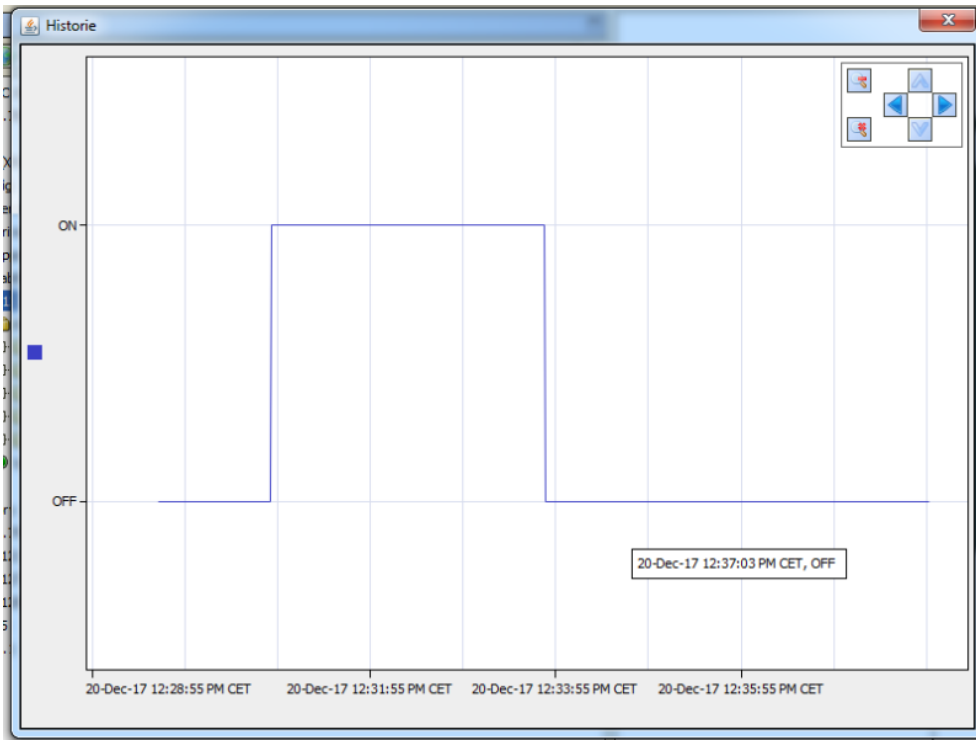
2.6 Sledování změn proměnných

Pomocí přídavku [12], který umožňuje sledovat změny stavu vybraného příkazu, lze zaznamenat hodnoty v průběhu času do grafu. Jsou možné dva přídavky pro sledování s intervalem nebo bez intervalu. S intervalem je možné nastavit, kdy se bude snímat hodnota zvoleného prvku v případě, že se zvolí bez intervalu tak se bude snímat vždy, když se změní hodnota.

Přídavek se dá získat jako *extension* v knihovně historie přes paletu. Při nalezení stačí vzít a přidat k sledované proměnné a povolit. Výslednou charakteristiku je možné zobrazit v navigačním menu ve složce historie (Obrázek 2.4). Nadále je možné graf a data exportovat do tvaru pdf, html a csv (formát pro excel).



Obrázek 2.3: Přídavek extension u proměnné



Obrázek 2.4: Získaná charakteristika z historie proměnné

3. DATABÁZOVÝ MODEL PRO UKLÁDÁNÍ MAR DAT

Cílem této bakalářské práce je vytvoření databázového modelu pro ukládání MaR dat z budovy areálu Technické 12. Budova T12 je dělena na několik bloků, které jsou označeny písmeny A až H. Model je tvořen v pomocném databázovém nástroji Case studio 2, jako entity-relationship diagram, ERD (*entitně-vztahový diagram*) a na základě tohoto modelu je pak realizována databáze v Microsoft SQL Serveru, která je propojena se softwarem COACH^{AX}.

Úvodem této kapitoly bych přiblížil relační databáze, dále se věnoval návrhu databázového modelu a v neposlední řadě realizací tohoto modelu.

3.1 Relační databáze

Relační databáze [14] využívají tabulky, které jsou propojeny předem nastavenými vztahy – relacemi. Tabulka obsahuje jednotlivé sloupce – atributy a řádky – záznamy. Atributy tabulky určují vlastnosti objektů, které se do tabulky budou vkládat. Do tabulky se vkládají objekty stejného druhu, nicméně každý vždy jen jednou. Atributy při návrhu tabulky neobsahují samotné hodnoty, pouze určují, jaké vlastnosti budou vkládané objekty mít v databázi uložené.

Atribut

Atribut je sloupec v tabulce, vlastnost záznamů v tabulce. U atributu určujeme při návrhu tabulky jeho doménu – rozsah hodnot, kterých může nabývat (doména může být například u atributu rozsah teploty od 0 do 100). Při vlastní implementaci (sestavení) navržené databáze do DBMS se pak z domény atributů určí jeho datový typ (například pro doménu výše je to Integer = celé číslo). DBMS se pak postará o to, že do daného sloupce nebude možné zapsat hodnotu, která nevyhovuje datovému typu.

Primární klíč

Tabulky nemají nijak pevně zafixováno pořadí záznamů – řádků, aby se tedy dalo jednoznačně ukázat na jeden záznam, pro editaci nebo jeho vybrání, je nutné zvolit jeden atribut tabulky jako unikátní. Takovému atributu říkáme primární klíč – každý záznam v tabulce musí mít hodnotu tohoto atributu unikátní pro celou tabulku, obvykle se jedná o celočíselné řady a každý novější záznam dostává číslo vždy o jednotku vyšší, než je číslo u posledního vloženého záznamu. Každá tabulka musí mít vytvořen primární klíč.

Cizí klíč

Záznamy v tabulkách, které jsou propojeny vztahy musí být nějakým způsobem provázány. Provázání se docílí vytvořením speciálního atributu – tzv. cizího klíče [14] – v jedné z tabulek. Do tohoto atributu se pak bude u jednotlivých záznamů kopírovat hodnota cizího klíče navázaného záznamu z druhé tabulky.

Relace

Mezi tabulkami existují čtyři druhy logických vztahů, ty určujeme při návrhu databáze, a jsou jimi:

Bez relace

Mezi daty v tabulkách není žádná spojitost, proto nedefinujeme žádný vztah.

Relace 1:1

Používáme, pokud záznamu odpovídá právě jeden záznam v jiné databázové tabulce a naopak. Takovýto vztah je používán pouze ojediněle, protože většinou není pádný důvod, proč takovéto záznamy neumístit do jedné databázové tabulky. Jedno z mála využití je zpřehlednění rozsáhlých tabulek.

Relace 1:N

Přiřazuje jednomu záznamu více záznamů z jiné tabulky. Jedná se o nejpoužívanější typ relace, jelikož odpovídá mnoha situacím v reálném životě. Jako reálný příklad může posloužit vztah VZT s místnostmi v budově T12 – jednotlivá VZT může obsluhovat více místností, ale jednu místnost nemůže obsluhovat více VZT.

Relace M:N

Umožňuje každému záznamu z jedné tabulky přiřadit libovolný počet záznamů z druhé tabulky, přičemž záznam v druhé tabulce přiřazením k záznamu k první tabulce svou možnost přiřazení „nespotřebuje“, takže jej lze přiřadit k libovolnému počtu záznamů první tabulky. V databázové praxi bývá tento vztah z praktických důvodů nejčastěji realizován kombinací dvou vztahů, a sice 1: N a 1:M, které ukazují do pomocné tabulky, složené z kombinace obou použitých klíčů (jde o třetí, resp. tzv. vazební tabulku).

3.2 Návrh databázového modelu

K návrhu modelu byly použity data z bloku H budovy T12, prvním krokem bylo tedy seznámení se s daty (Obrázek 3.1), všechna data jsou k nahlédnutí v přiloženém dokumentu na CD (Příloha 1).

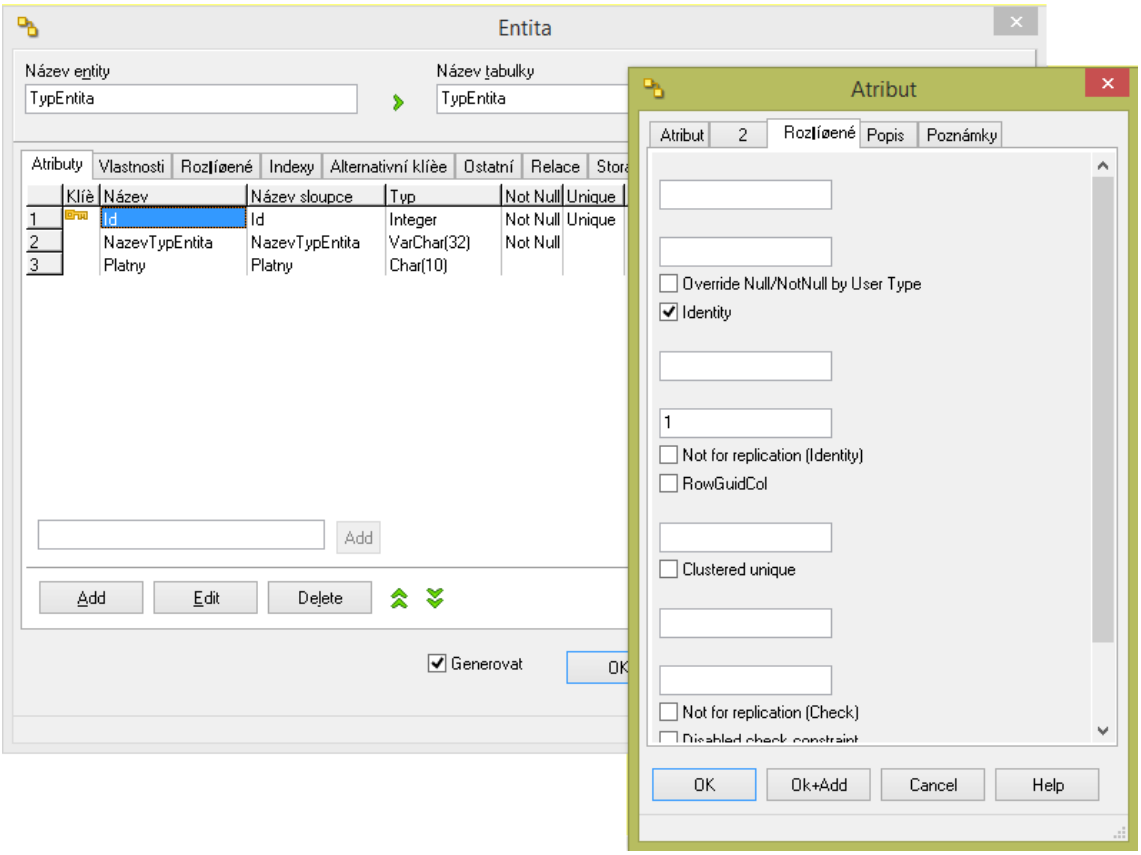
	A	J	W	X
1	ItemName	ItemDescription	EXCEL5000PointPointEnterpriseName	EXCEL5000PointPo
23	T12_H_MaR_VZ01_EmSpot	H, MaR VZT-01, EMAX spotřeba celého zařízení v KWh	T12_H_MaR_VZ01_EmSpot	PointValue
24	T12_H_MaR_VZ01_FM_od	H, MaR VZT-01, Frekvenční měnič vent odtah	T12_H_MaR_VZ01_FM_od	PointValue
25	T12_H_MaR_VZ01_FM_pr	H, MaR VZT-01, Frekvenční měnič vent přívod	T12_H_MaR_VZ01_FM_pr	PointValue
26	T12_H_MaR_VZ01_Ch_V_od	H, MaR VZT-01, Chod (dP) ventilátor odtah	T12_H_MaR_VZ01_Ch_V_od	PointState
27	T12_H_MaR_VZ01_Ch_V_pr	H, MaR VZT-01, Chod (dP) ventilátor přívod	T12_H_MaR_VZ01_Ch_V_pr	PointState
28	T12_H_MaR_VZ01_Ch_CH1	H, MaR VZT-01, Chod zdroje chladu	T12_H_MaR_VZ01_Ch_CH1	PointState
29	T12_H_MaR_VZ01_Ch_CH2	H, MaR VZT-01, Chod zdroje chladu	T12_H_MaR_VZ01_Ch_CH2	PointState
30	T12_H_MaR_VZ01_Pritom1154	H, MaR VZT-01, Karta přítomnosti	T12_H_MaR_VZ01_Pritom1154	PointState
31	T12_H_MaR_VZ01_Klap_cir	H, MaR VZT-01, Klapka cirkulace	T12_H_MaR_VZ01_Klap_cir	PointValue
32	T12_H_MaR_VZ01_Klap	H, MaR VZT-01, Klapka přívod a odtah	T12_H_MaR_VZ01_Klap	PointState
33	T12_H_MaR_VZ01_Kor_1154	H, MaR VZT-01, Korekce	T12_H_MaR_VZ01_Kor_1154	PointValue
34	T12_H_MaR_VZ01_CO2_1154	H, MaR VZT-01, Měření kvality vzduchu podle CO2 v prostoru	T12_H_MaR_VZ01_CO2_1154	PointValue
35	T12_H_MaR_VZ01_CO2_%	H, MaR VZT-01, Měření kvality vzduchu podle CO2 v%	T12_H_MaR_VZ01_CO2_%	PointValue
36	T12_H_MaR_VZ01_Mix_oh	H, MaR VZT-01, Mix ohřev	T12_H_MaR_VZ01_Mix_oh	PointValue
37	T12_H_MaR_VZ01_Por_c_oh	H, MaR VZT-01, Porucha čerpadla ohřev	T12_H_MaR_VZ01_Por_c_oh	PointState
38	T12_H_MaR_VZ01_Por_FMod	H, MaR VZT-01, Porucha frekvenčního měniče	T12_H_MaR_VZ01_Por_FMod	PointState
39	T12_H_MaR_VZ01_Por_FMpr	H, MaR VZT-01, Porucha frekvenčního měniče	T12_H_MaR_VZ01_Por_FMpr	PointState

Obrázek 3.1: Ukázka dat budovy H

Po nastudování dokumentu s daty jsem v nástroji CASE studio 2, začal s návrhem databázového modelu. Model je tvořen, tak aby vyhovoval všem blokům budovy T12, popřípadě takto koncipované budově.

Ze začátku návrhu jsem vytvořil tabulky, které jsou pojmenovaným podle sdružených dat. Pro celý návrh – názvy tabulek, sloupců a další identifikátory jsem použil notaci CamelCase (první písmeno slova velké, další malá, bez mezer) pro větší přehlednost v následné realizaci databáze. Celý navrhovaný databázový model obsahuje celkem šest tabulek viz (Obrázek 3.3).

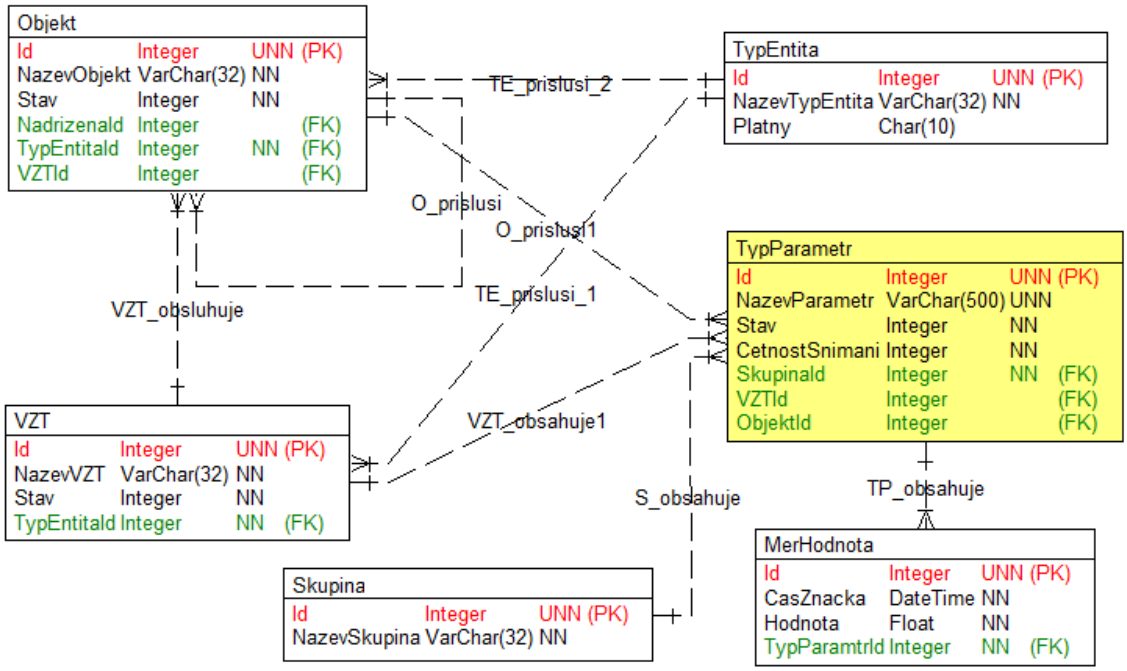
Základní tabulka nese název *TypEntita*, která by měla obsahovat data sdružující celek např. areál, blok apod. Tabulka *TypEntita* má definované tyto atributy: *Id, NazevTypEntita* a *Platny*. Výhodou nástroje CASE studio 2 je předdefinování celé databáze, takže při návrhu už přemýšlíme, jak přesně by měly být definovány primární klíče jednotlivých tabulek a jaké datové typy budou jednotlivé atributy mít. Pro každou vytvořenou tabulku jsem zvolil primární klíč s názvem *Id* datového typu Integer – celé číslo. Ve vlastnostech takto vytvořeného atributu jsem zvolil, aby byl *unique* (měl vždy unikátní hodnotu, nemohlo se stát, že by se objevoval v databázi více než jednou), *not null* (měl vždy nějakou hodnotu, v našem případě celé číslo) a v neposlední řadě jsem zaškrtl prvek *Identity* a do prázdné kolonky zapsal jedničku (toto nám zaručuje, že nově přidaná hodnota bude vždy o jedno číslo větší než předchozí) viz (Obrázek 3.2).



Obrázek 3.2: Automatická inkrementace

Atribut *NazevTypEntita* jsem určil jako VarChar (Variable Character Field), označení textového řetězcového datového typu a na rozdíl od klasického typu Char zabírá proměnlivou velikost paměti, dále jsem určil vlastnost *not null*. Takto zvolený sloupec bude vyplňovat uživatel nebo nějaký automatizovaný proces a jak už bylo zmíněno dříve, určujeme tímto popis celku dat. Posledním atributem v tabulce *TypEntita* je atribut *Platny*, který je datového typu Char a nese údaj o stavu dat ve vyplněné tabulce.

Další tabulkou v návrhu databáze je tabulka s názvem *VZT*, která slouží pro shromažďování názvů vzduchotechnických zařízení a obsahuje atributy: *Id, NazevVZT, Stav* a *TypEntitaId*. Atributy *Id* a *NazevVZT* jsou obdobné jako atributy v tabulce *TypEntita*. Další atribut *Stav* je datového typu Integer a nese údaj v jakém stavu se data v tabulce nachází. Poslední atribut, v tabulce *VZT*, *TypEntitaId* je takzvaně cizí klíč, který nám propojuje pomocí relace data z tabulky *TypEntita* s tabulkou *VZT* vazbou 1:N. Tento atribut má vlastnost *not null*, protože vždy bude spadat pod nadřízená data z tabulky *TypEntita*.



Obrázek 3.3: Model databáze

Třetí tabulka návrhu databázového modelu, *Objekt*, se od předchozí tabulky *VZT* liší v počtu cizích klíčů, jinak má atributy obdobné: *Id*, *NazevObjekt*, *Stav*, *NadrizenaId*, *TypEntitaId* a *VZTId*. Tato tabulka bude shromažďovat data na úrovni názvů jednotlivých bloků a místností v budově T12. Cizí klíče, jako např. *Nadrizena* nám určuje pomocí selfrelace odkaz zpět do tabulky na nadřízená data a zlepšuje přehlednost takových dat, tento cizí klíč není *not null*, protože se může stát, že nemá v této tabulce nadřízená data. Dalším cizím klíčem v tabulce *Objekt* je *TypEntitaId*, který hraje stejnou roli jako atribut v tabulce *VZT*. Posledním cizím klíčem je poté *VZTId*, který odkazuje do tabulky *VZT* na nadřízenou vzduchotechniku. Takto zvolený atribut není *not null*, protože se může stát, že pro celek jako T12 nemůžeme zapsat žádnou nadřízenou vzduchotechniku, ale pro jednotlivou místnost už ano.

Čtvrá tabulka, tabulka *MerHodnota*, je hlavní tabulkou pro zápis hodnot MaR dat. Obsahuje atributy: *Id*, *CasZnacka*, *Hodnota* a *TypParametrId*. *Id*, jak už bylo zmíněno, je primární klíč této tabulky. Atribut *CasZnacka* je datového typu DateTime, který slouží k ukládání dat představujících datum nebo čas ve formátu (YYYY-MM-DD HH:MM:SS) a tvoří časovou značku uložené hodnoty MaR dat, a proto je nastaven vlastností *not null*. Dále tabulka *MerHodnota* obsahuje atribut *Hodnota* datového typu Float (racionální číslo) pro zapsání hodnoty MaR dat, taktéž je tento atribut nastaven jako *not null*. V neposlední řadě nám zbývá atribut v podobě cizího klíče. Tabulka *MerHodnota*, co se týče hiearchie, je na nejnižším stupni, tedy má nad sebou nadřazená data, se kterými je spojena pomocí relace. Tudíž cizí klíč *TypParametrId* odkazuje na

tabulku *TypParametr* a je nastaven příznakem *not null*, protože vždy bude mít měřená hodnota nadřazené informace. Tyto informace obsahují tabulky *TypParametr* a *Skupina*.

Tabulka *Skupina* nese informace o skupině měřených dat, např. taková data by mohly nést název MaR. Tato tabulka má dva atributy, v podobě primárního klíče *Id* a textového datového typu *NazevSkupina*.

Tabulka *TypParametr* nese informace, jak často se měřená hodnota bude snímat, její název a stav, udávající, v jakém životním cyklu se nachází. Atributy této tabulky jsou: *Id, NazevParametr, Stav, CetnostSnimani, SkupinaId, VZTId, ObjektId*.

Takto vytvořený model je ve finální podobě a připraven pro realizaci databáze. Této problematice se budu věnovat v následující části kapitoly.

3.3 **Realizace databázového modelu**

Pro realizaci databázového modelu byl zvolen software Microsoft SQL Server (MS-SQL). Microsoft SQL Server je relační databázový a analytický systém pro řešení datových skladů vyvinutý společností Microsoft.

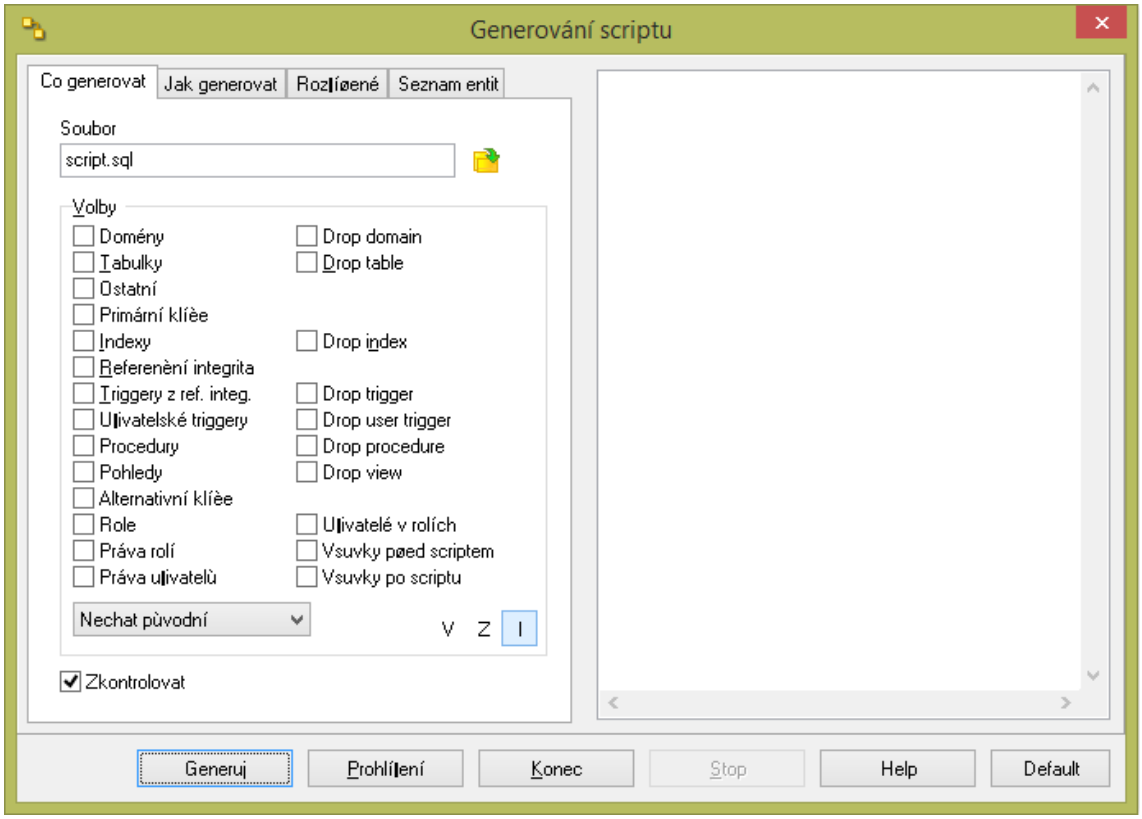
Označení databáze je vlastně nepřesné a v odborné literatuře se setkáme s označením RDBMS [15] (Relation DataBase Management System). Česky je to přeloženo jako "systém řízení báze dat". Databázový stroj (tedy zde MS-SQL) není jen úložiště dat. Jedná se o velmi sofistikovaný a odladěný nástroj, který za nás řeší spoustu problémů a zároveň je extrémně jednoduchý k použití. S databází totiž komunikujeme jazykem SQL, kterým jsou v podstatě lidsky srozumitelné věty. Spolu s ukládáním dat je, ale třeba dále řešit mnoho dalších věcí. Asi by nás napadlo např. zabezpečení nebo optimalizace výkonu. RDBMS toho, ale dělá ještě mnohem více, řeší za nás problém současné editace stejné položky několika uživateli ve stejný okamžik, který by jinak mohl zapříčinit nekonzistenci databáze. RDBMS data v tomto případě zamkne a odemkne až po vykonání zápisu. Dále umožňuje spojovat několik dotazů do transakcí, kdy se série dotazů vykoná vždy celá nebo vůbec. Nestane se, že by se vykonala jen část. Tyto vlastnosti databázového stroje jsou shrnovány zkratkou ACID. ACID je akronym slov *Atomicity* (nedělitelnost), *Consistency* (validita), *Isolation* (izolace) a *Durability* (trvanlivost).

Databáze (přesněji databázový stroj) je tedy černá skříňka, se kterou naše aplikace komunikuje a do které ukládá veškerá data. Její použití je velmi jednoduché a je odladěna tak, jak bychom si sami zápis dat v programu asi těžko udělali. Vůbec se nemusíme starat o to, jak jsou data fyzicky uložena, s databází komunikujeme pomocí jednoduchého dotazovacího jazyka SQL, viz dále. O databázi občas hovoříme jako o 3. vrstvě aplikace (1. vrstva je uživatelské rozhraní, 2. vlastní logika aplikace, 3. je právě datová vrstva).

V našem případě bylo zvoleno uživatelské rozhráním za pomocí softwaru COACH^{AX}.

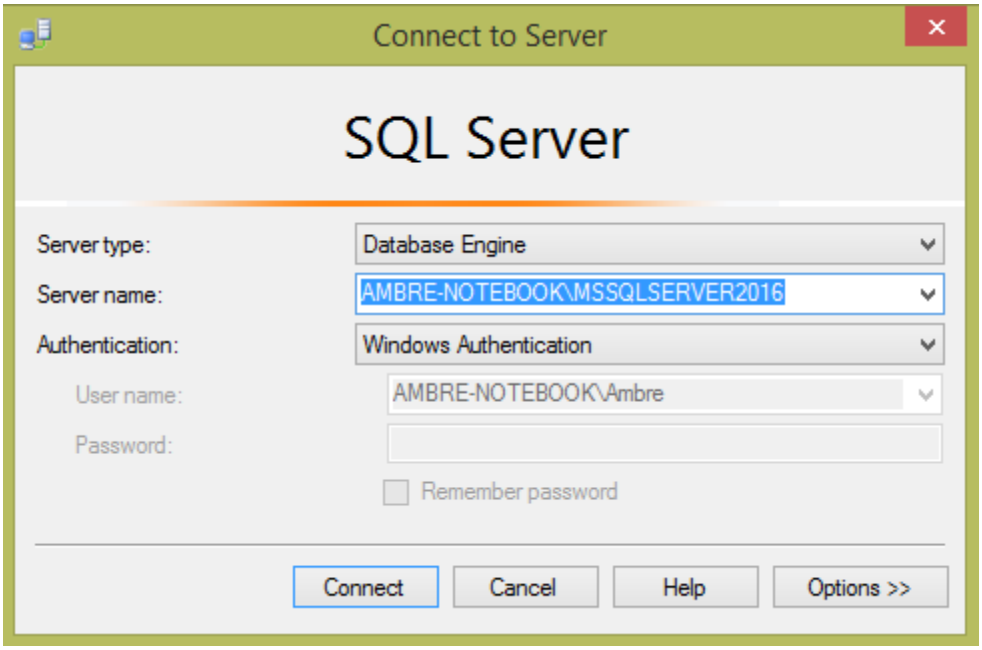
3.3.1 Tvorba databáze v MS-SQL

Pro tvorbu databáze v MS-SQL použijeme automatické vygenerování skriptu z pomocného nástroje CASE studio 2, který si následně poupravujeme podle potřeby v MS-SQL. Vygenerování skriptu provedeme tak, že v návrhovém studiu CASE studio 2 zvolíme v horní liště *Model* a dále vybereme položku *Generování scriptu*. Tento proces můžeme vykonat i pomocí klávesy *F9*. Následně nám vyskočí okno viz (Obrázek 3.4). Zde v kolonce pod *Soubor* zvolíme název našeho skriptu a pomocí žluté ikonky vybereme cestu kam se nám uloží. Dále zaškrtneme, co přesně chceme generovat a zmáčkneme v levém dolním rohu na *Generuj*. Při případné chybě nám vyskočí okno, kde je chyba popsána, a my ji po přečtení můžeme vyřešit. Když nám žádné okno s chybou nevyskočilo můžeme v pravé polovině okna *Generování scriptu* vidět průběh generování. Při úspěšném dokončení můžeme celý pomocný nástroj Case studio 2 zavřít a spustit software MS-SQL Server.



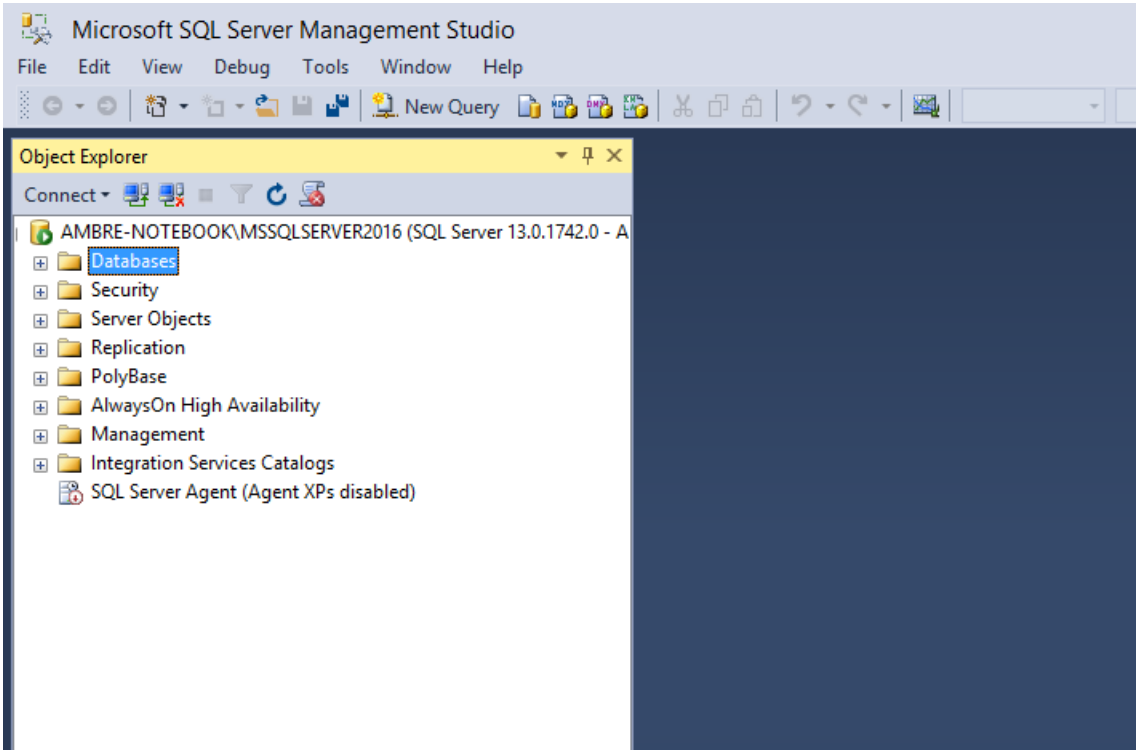
Obrázek 3.4: Generování skriptu

Při spuštění MS-SQL Serveru budeme požádáni o přihlášení, viz (Obrázek 3.5). Zde vybereme v kolonce *Server name* jméno serveru, kde bude databáze uložena a k autentizaci vybereme stejné přihlašovací údaje jako máme k přihlášení do systému Windows. V mém případě se tato varianta zdála jako chybná a musel jsem použít k autentizaci *SQL Server Authentication*, vyplnit *User name* a *Password*. Bohužel i nadále se do databáze nedařilo přihlásit a po soustavném bádání jsem došel k závěru, že nemám práva k přístupu od administrátora. Požádal jsem tedy administrátora o práva k přístupu a připojil se k databázi.



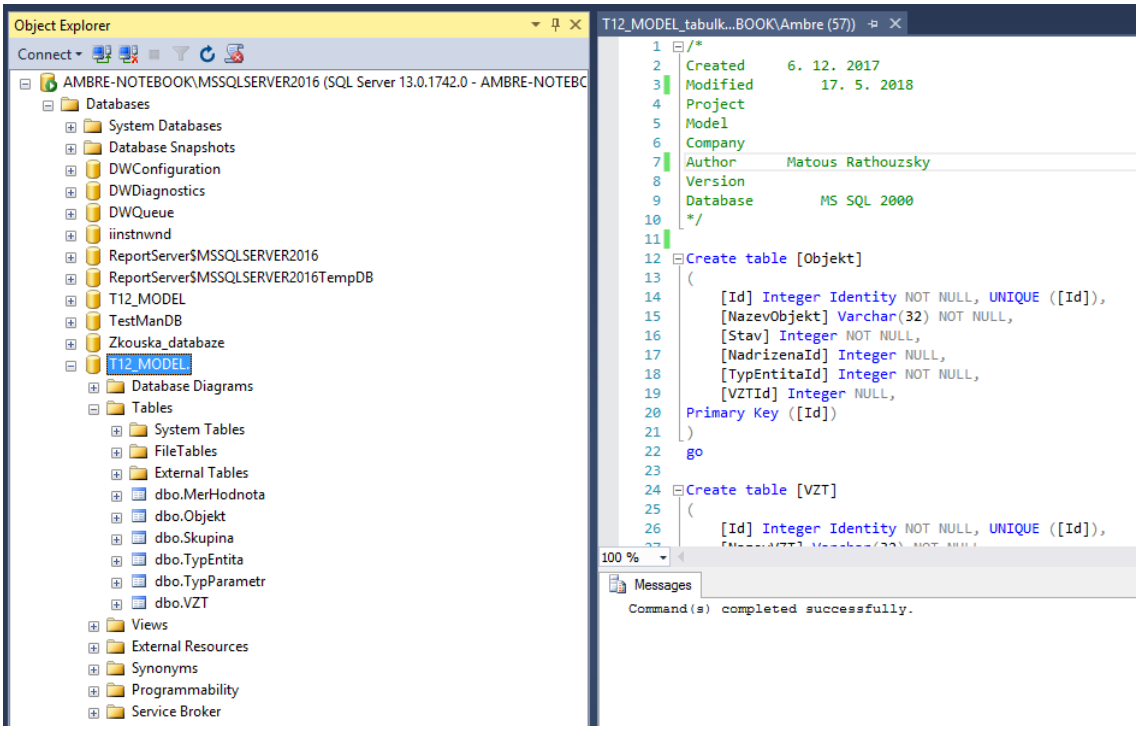
Obrázek 3.5: Přihlášení MS-SQL

Po přihlášení do MS-SQL vidíme v levé části okno *Object Explorer* a náš úložný prostor viz (Obrázek 3.6). Můžeme tedy začít s vytvořením naší databáze. Klikneme v tomto okně pravým tlačítkem myši na položku *Database* a zvolíme *New Database*. Následně nám vyskočí okno pro vytvoření databáze, zvolíme její název, v mém případě „T12_MODEL“ a potvrdíme *OK* pro vytvoření. Při úspěšném vytvoření naši vytvořenou databázi můžeme vidět po rozkliknutí položky *Database*. Takto vytvořená databáze by zatím měla být prázdná. V následujícím kroku to změníme.



Obrázek 3.6: MS-SQL – Object Explorer

Nyní do popředí přichází náš vygenerovaný skript z pomocného nástroje CASE studio 2. Tento skript si za pomocí MS-SQL otevřeme. Kliknutím v levém horním rohu na položku *File*, následně *Open* a *File* nebo zrychleně pomocí klávesové zkratky *CTRL*+O, následně zvolíme jeho umístění. Po otevření skriptu vidíme zapsaný kód v jazyce SQL. V horní liště klikneme na *Execute* nebo zmáčkneme klávesu *F5* pro provedení tohoto kódu. Následně bychom měli obdržet hlášku *„Command(s) completed successfully.“* a po rozkliknutí naší databáze *T12_MODEL* vidět vytvořené tabulky, viz (Obrázek 3.7).



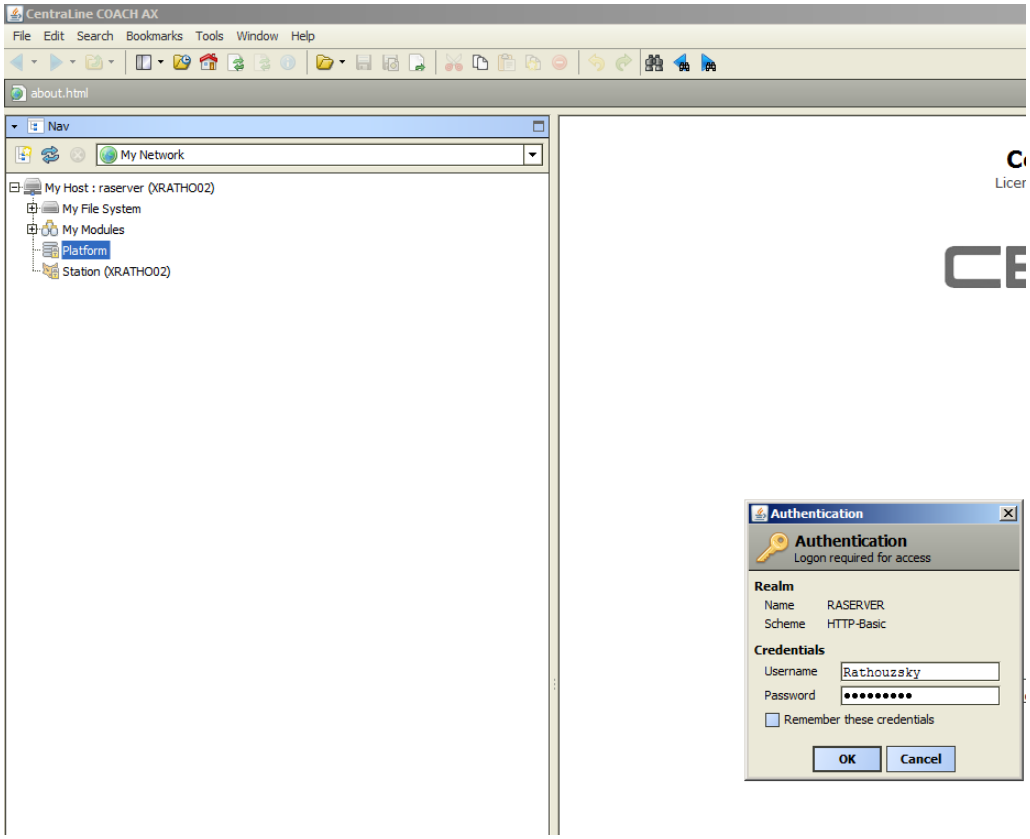
Obrázek 3.7: Použiti skriptu – generování tabulek

3.3.2 Připojení databáze k softwaru COACH^{AX}

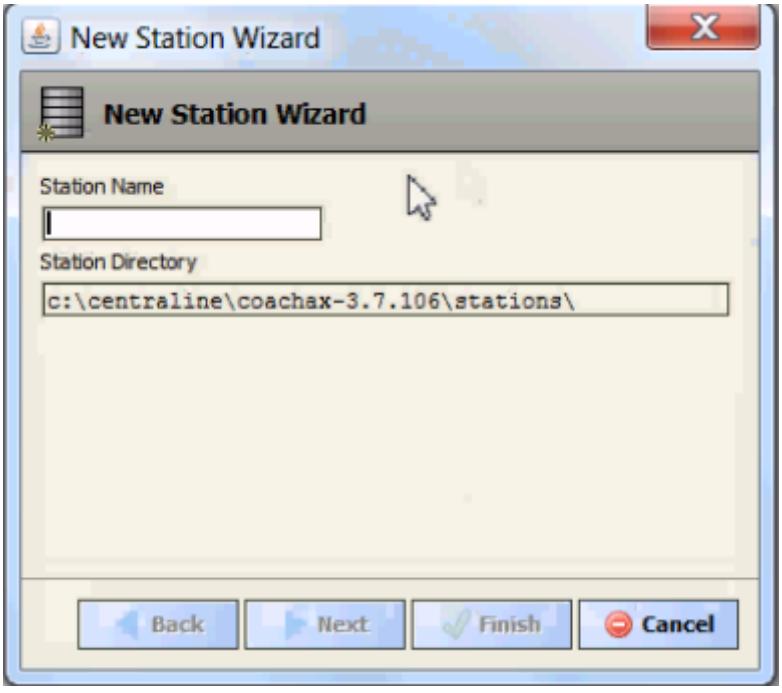
Po vytvoření databáze v MS-SQL je potřeba ji nějakým způsobem obsluhovat. Obsluhovat databázi budeme pomocí softwaru COACH^{AX}.

Prvotním krokem po startu softwaru COACH^{AX} je připojení do platformy, připojení najdeme v horní liště pod položkou *File*, *Open* a *Open platform* nebo pomocí klávesové zkratky *Ctrl+shift+P*, zde poté musíme vyplnit IP adresu a objeví se nám okno pro přihlášení (Obrázek 3.8). Vyplníme přihlašovací údaje, v mém případě *Login* a *Heslo* je shodné s údaji do přihlášení k operačnímu systému Windows. Po přihlášení do platformy si vytvoříme stanici. Vytvoření stanice najdeme pod položkou *Tools* v horní liště COACH^{AX} jako *New Station* a po vyskočení okna (Obrázek 3.9) zadáme její název a umístění. Klikneme na tlačítko *Next* vyplníme heslo a potvrdíme tlačítkem *Finish*.

Takto vytvořenou stanici připojíme k platformě, otevřeme platformu v levé liště COACH^{AX} rozklikneme a dále rozklikneme *Application Director*, zde označíme naší stanici a spustíme jí tlačítkem *Star*. Dále se do naší stanice připojíme, to provedeme obdobně jako připojení platformy pod položkami *File*, *Open* a *Open Station* nebo klávesovou zkratkou *Ctrl+Shift+O*. Zde zadáme IP adresu a po vyskočení okna přihlášení k stanici zadáme do položky *Username* „admin“ a *Password* zvolené heslo.

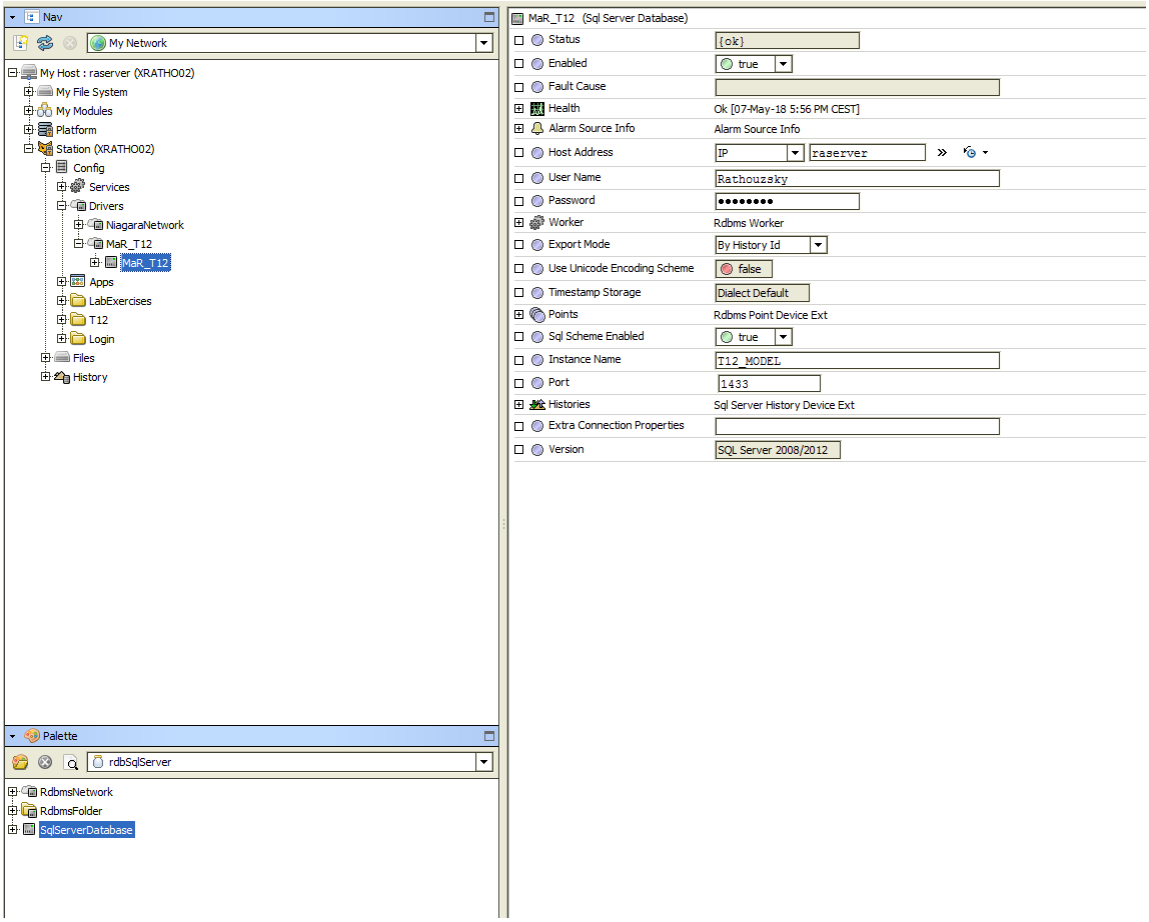


Obrázek 3.8: Přihlášení platformy v COACH^{AX}



Obrázek 3.9: COACH^{AX} – New Station

V tuto chvíli nám už nic nebrání k propojení databáze MS-SQL se softwarem COACH^{AX}. V teoretickém úvodu byl popsán prvek *Palette* v COACH^{AX}, který v tomto okamžiku využijeme. V dialogovém okně *Palette* vyhledáme *rdbSqlServer* a přesuneme prvek *RdbmsNetwor* do adresáře *Drivers* naší stanice. Pojmenujeme ho jako „MaR_T12“ a z *Palette* do něj vložíme další prvek *SqlServerDatabase*, který pojmenujeme taktéž „MaR_T12“, následně jej otevřeme pro nastavení připojení k databázi MS-SQL viz (Obrázek 3.10).



Obrázek 3.10: Připojení databáze k stanici COACH^{AX}

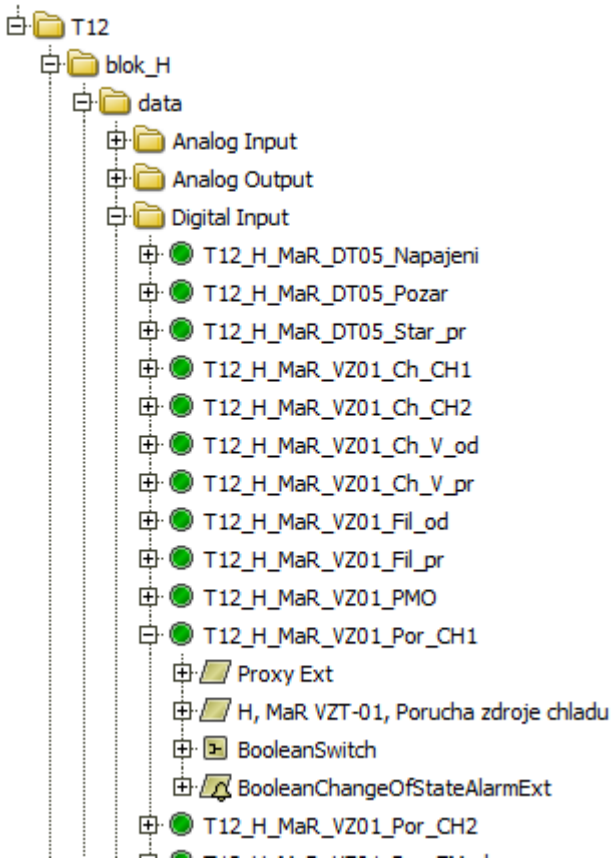
V nastavení připojení vyplníme IP adresu databáze MS-SQL, naše přihlašovací údaje k ní a v položce *Instance Name* zadáme jméno vytvořené databáze v MS-SQL. Vyplněné údaje uložíme pomocí dolního tlačítka *Save* a ověříme komunikaci mezi COACH^{AX} a databází pomocí příkazu *ping*, která je dostupná po kliknutí pravým tlačítkem myši na prvek *MaR_T12* pod položkou *Actions*. Při správném průběhu by se nám mělo objevit v nastavení připojení databáze v kolonce *Status* „ok“.

3.3.3 Obsluha databáze

Představou softwaru COACH^{AX} při obsluze databáze MS-SQL bylo použití formulářů, kde by si správce/uživatel zadal jaká data bude do databáze zapisovat, popřípadě číst z databáze. Bohužel po ustavičné práci se softwarem COACH^{AX} vyplynulo, že taková realizace nebude možná. Vše se bude muset dělat tedy ručně nastavením pro každou proměnnou zvlášť a pomocí příkazů SQL. Následující článek bude popisovat průběh takového procesu.

Administrátor nebo vývojář stanice už při prvotním kroku vytváření proměnných, které budou reprezentovat data z budovy, musí založit logiku, jak taková data propojit. V našem případě jsem vytvořil ve stanici složku *data* viz (Obrázek 3.11), kde pod jednotlivými podsložkami jsou proměnné týkající se MaR dat budovy T12. Tyto proměnné jsou napojeny přes *switch* pro reálná nebo simulovaná data, viz Kapitola 4.1.

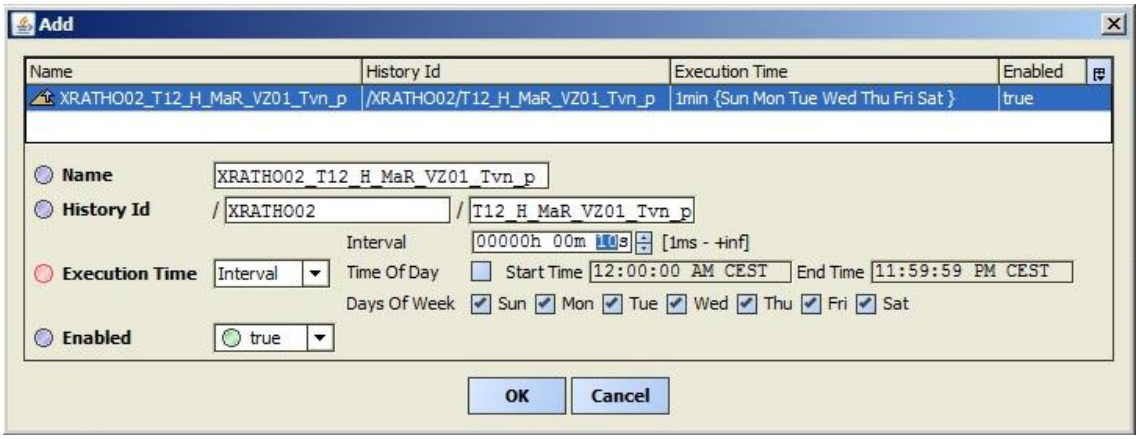
Každá proměnná má také připojeno rozšíření *extension* pro zachování její historie a díky tomuto rozšíření můžeme proměnnou zapisovat do naší databáze. Takto zapsaná data v databázi vytvoří samostatnou tabulku od softwaru COACH^{AX} se kterou budeme následně za pomocí příkazů SQL pracovat.



Obrázek 3.11: Proměnné reprezentující MaR data budovy T12

3.3.4 Ukládání dat ze softwaru COACH^{AX} do MS-SQL

Pro ukládání dat do databáze rozklikneme prvek *MaR_T12*, zde otevřeme *Histories* a klikneme na tlačítko *Discover*. Okno *Histories* se nám rozdělí na dvě části – horní *Discovered*, dolní *Database*. V horní části rozklikneme nabízený prvek a uvidíme rozšíření *extension* každé proměnné, kterou jsme vytvořili. Vybereme si jednotlivou proměnnou dvojklikem, následně se nám zobrazí okno *Add* (Obrázek 3.12). V tomto okně zadáme pod položkou *Execution Time* Interval pro export dat do databáze MS-SQL, klikneme na tlačítko *OK* a následně můžeme vidět v dolní části *Histories*, takto přidanou proměnnou pro export dat do databáze. Pro kontrolu, jestli vše proběhlo správně se podíváme do naší databáze v MS-SQL. V databázi *T12_MODEL* bychom pod položkou *Tables* měli najít dvě nové tabulky *dbo.HISTORY_CONFIG* a dbo.*XRATHO02_T12_H_MAR_VZ01_TVN_P*. Tento proces opakujeme pro všechny proměnné, které chceme mít uložené v databázi.



Obrázek 3.12: Ukládání proměnné do databáze

3.3.5 Třídění dat do předpřipravených tabulek

Přichází na řadu ručně psané kódy za pomocí příkazů SQL, a proto bych na úvod tento jazyk trochu přiblížil.

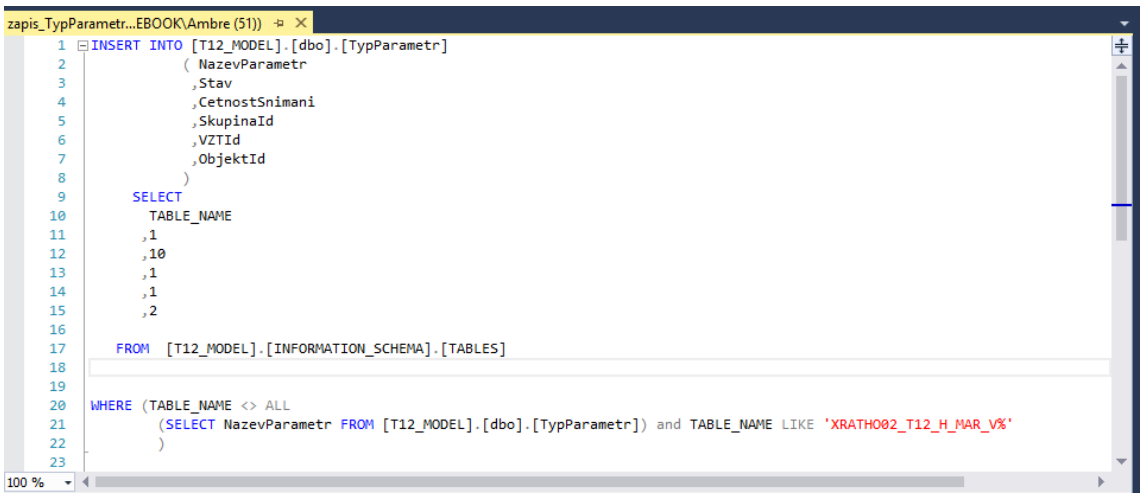
SQL [16] – *Structured Query Language* (Strukturovaný dotazovací jazyk) - je především interaktivní dotazovací jazyk – umožňuje získat odpovědi i na velmi komplikované dotazy téměř ihned. Je nástrojem neprocedurálním, s množinovým přístupem k datům a je jazykem standardizovaným, je srozumitelný, protože chápe data v podobě tabulek, což je snadno pochopitelné i uživatelům.

Příkazy jazyka SQL by se daly rozdělit do těchto skupin [17]:

- DDL (*data definition language*) - příkazy patřící do této skupiny vytvářejí či upravují strukturu databáze (např. tabulky). Příklady: CREATE, ALTER, DROP...
- DML (*data manipulation language*) - příkazy, které slouží k získávání, ukládání a mazání dat v databázi. Příklady: SELECT, INSERT, UPDATE, DELETE...
- DCL (*data control language*) - příkazy pro správu uživatelských rolí a práv. Příklady: GRANT, REVOKE...
- TCL (*transactional control language*) - příkazy pro správu databázových transakcí. Příklady: BEGIN, COMMIT, ROLLBACK...

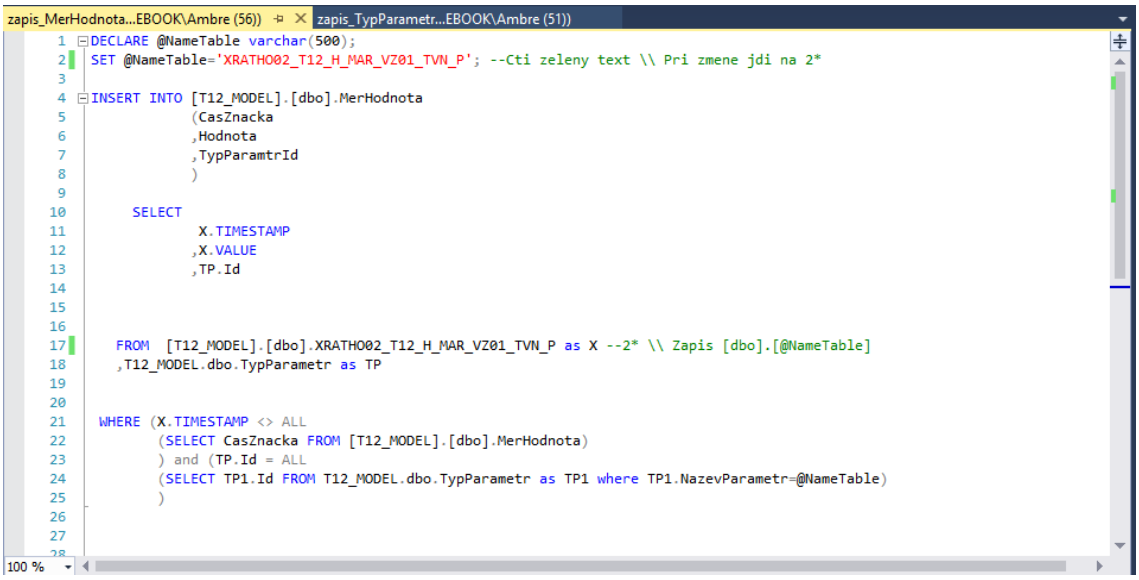
V našem případě použijeme příkazy DML. Třídění dat v databázi MS-SQL probíhá v následujícím sledu událostí, za pomocí poloautomatického procesu.

V prvotním kroku je zapotřebí napsat skript pro přečtení tabulek v databázi a vybrat jen ty, které jsou nově přidány ze softwaru COACH^{AX}. Z MS-SQL jsme zjistili jakým způsobem se přijde na názvy tabulek v naší databázi a tento proces jsme použili v našem skriptu. Protože nově vytvořené tabulky ze softwaru COACH^{AX} mají vždy unikátní název, a začínají názvem *XRATHO02_T12_*, použijeme tento fakt k vytřídění tabulek týkajících se softwaru COACH^{AX}. Dále do námi vytvořených tabulek ručně zapíšeme hodnoty nadřízených pro tabulku *TypParametr* a zjištěné názvy nově přidaných tabulek ze softwaru COACH^{AX} přidáme do tabulky *TypParametr* jako *NazevTypParametr* a propojíme cizími klíči na nadřízené tabulky. Celý skript bude pak vypadat následovně viz (Obrázek 3.13).



Obrázek 3.13: Třídění dat – zápis TypParametr

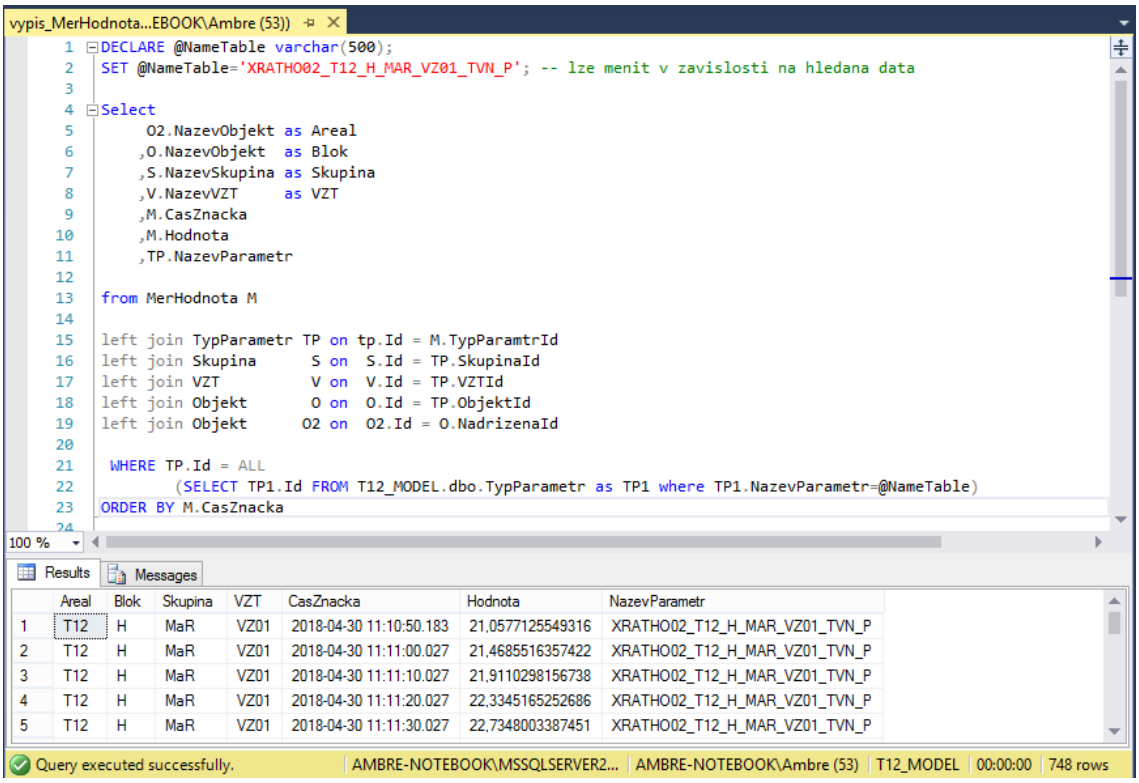
Dalším krokem je vytvoření skriptu pro zápis dat do tabulky *MerHodnoty* a propojit je se správným atributem *NazevTypParamtr*. Takto vytvořený skript začíná deklarováním proměnné pro zjednoduší práci. Bohužel jsem, ale při vývoji skriptu zjistil, že funkcionalitu deklarované proměnné nevyužiji podle mých představ v celém skriptu a za SQL příkazem *WHERE* se musí obsah proměnné ručně zapsat. Celý skript tkví na použití SQL příkazu *INSERT INTO*, za který vybereme tabulku a sloupce z ní, do kterých se bude zapisovat. Dále příkazem *SELECT* vybereme zapisovaná data a příkazem *FROM* řekneme odkud tyto data brát. Jako poslední pak pomocí příkazu *WHERE* vytvoříme logiku, aby se opravdu vybrala jen ty zapisovaná data, která si přejeme. Celý skript bude pak vypadat následovně viz (Obrázek 3.14).



Obrázek 3.14: Třídění dat – měřená data

3.3.6 Výpis dat z databáze MS-SQL

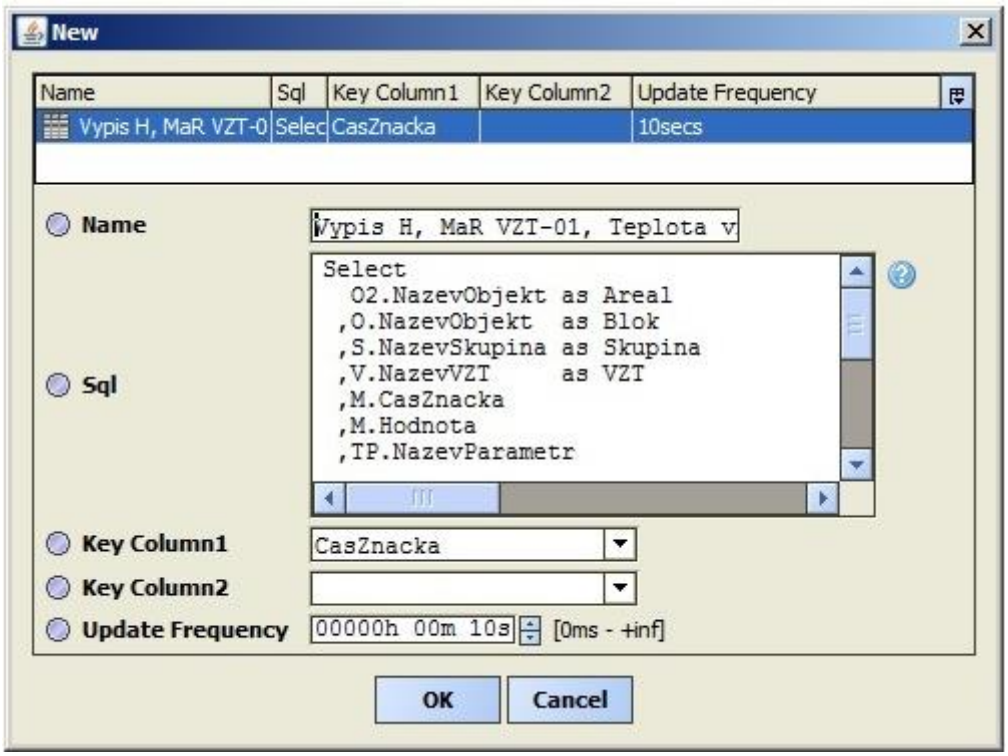
První krokem při výpisu dat bude zkouška výpisu dat za pomocí vytvořeného skriptu v MS-SQL a následně tento výpis aplikovat do softwaru COACH^{AX}. K výpisu dat z databáze použijeme již známý příkaz *SELECT* a malou logiku k propojení dat mezi tabulkami, viz (obrázek 3.15).



Obrázek 3.15: Ukázka výpisu dat v MS-SQL

Na začátku skriptu si deklarujeme proměnnou *NameTable* pro univerzální vyhledání dat. Dále za příkazem *SELECT* vybereme sloupce, které nás z tabulek zajímají a budeme je chtít zobrazit ve výpisu dat. Příkaz *AS* slouží jako *alias* (přezdívka). Dalším krokem vytvoříme logiku k propojení zobrazených dat za pomocí nadefinovaných relací. Tento proces provedeme příkazem *left join* za který napíšeme název tabulky, její zkratku pro jednoduší použití, a následně propojíme primární klíč s cizím klíčem. Takto vybraná propojená data omezíme příkazem *WHERE* pro zobrazení dat týkajících se jedné proměnné a seřadíme, podle času zapsání, příkazem *ORDER BY*. Celý sepsaný script následně spustíme, ověříme, že byla zobrazena námi chtěná data, pokud ano použijeme skript v softwaru COACH^{AX}.

Po ověření v MS-SQL se přesuneme do softwaru COACH^{AX}, kde vybereme položku *Points* nacházející se pod námi přidaný prvek *MaR_T12*. Klikneme na tlačítko *New* pro otevření okna *New* (Obrázek 3.16). Zde zvolíme název, vložíme přichystaný skript, do *Key Column1* zapíšeme *CasZnacka*, zvolíme 10s pro aktualizaci dat z databáze MS-SQL a potvrdíme *OK*. Následně se pod položkou *Points* vytvoří námi pojmenovaný prvek pro výpis dat. Prvek rozklikneme a uvidíme vypsaná data z databáze *T12_MODEL* proměnné *XRATHO02_T12_H_MAR_VZ01_TVN_P (Teplota vnitřní průměr)* obdobně jako v MS-SQL.

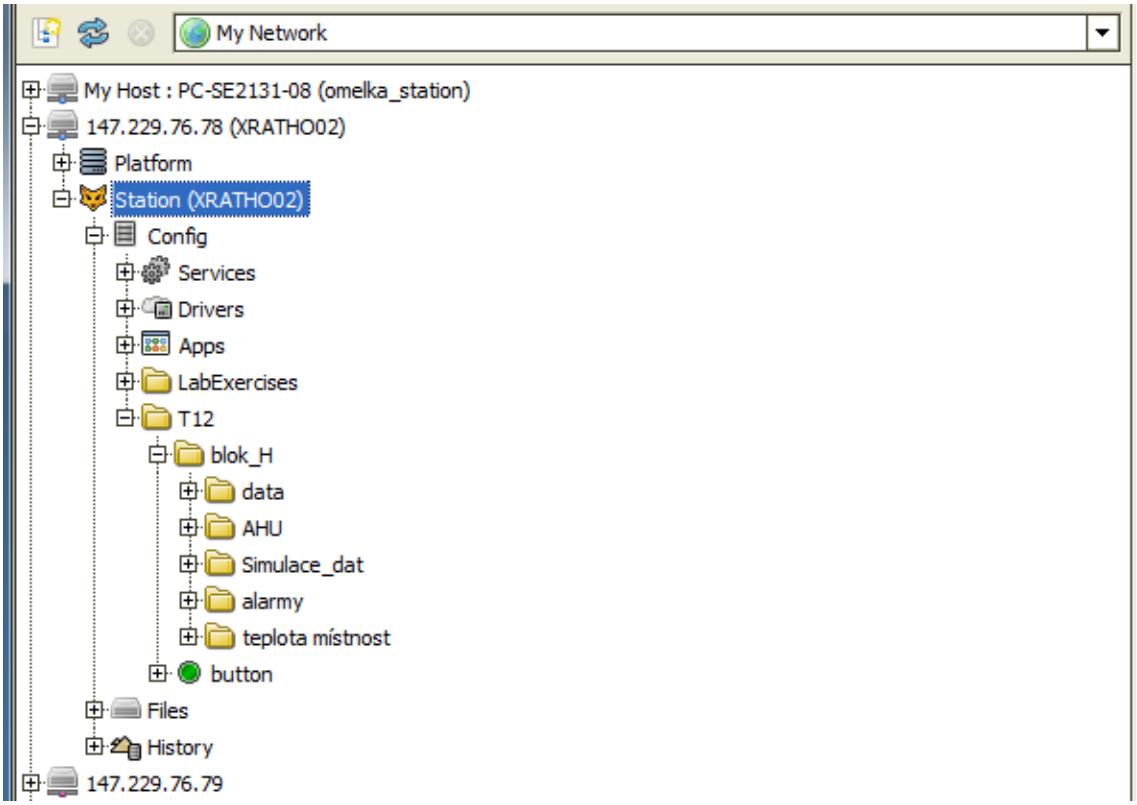


Obrázek 3.16: Výpis dat v softwaru COACH^{AX} z databáze MS-SQL

4. VIZUALIZACE A GENEROVÁNÍ MAR DAT

Vizualizace a generování MaR dat bylo prováděno v softwaru COACH^{AX} za použití vytvořené stanice z kapitoly 3.

Prvotním krokem, v této části práce, bylo vytvoření podsložek viz (Obrázek 4.1) pro jednoduší orientaci při vývoji. Dále byla do každé složky umístěna logika týkající se MaR dat budovy T12 bloku H a následně vytvořena vizualizace. Koncept vizualizace je založen na již existující vizualizaci [18] pana Podolského z roku 2017.



Obrázek 4.1: Náhled vytvořené stanice v softwaru COACH^{AX}

4.1 Generování dat

Složka *data* obsahuje všechny proměnné rozdělené podle vstupu, výstupu, zdali jsou analogové nebo digitální a dále se také používají k vizualizaci. Proměnné již rozšířeny o položku *history* umožňují sledování průběhu v čase, přepínač na reálná data a v případě proměnných týkajících se poruch byly tyto proměnné rozšířeny i o alarmy, které reagují na změnu hodnoty.

Složka *simulace dat*, je srdcem celého generování dat. Generují se zde signály a paralelně jsou převedeny na proměnné ve složce *data*, která se posléze používají. Pro

generování dat je zde vytvořeno několik pomocných proměnných datového typu *Boolean* – pro provozní dny, poruchy, léto/zima a typu *Numeric* – pro teploty, počet lidí v místnosti. Následně jsou tyto proměnné používány k vytvoření logické vazby tvořené na listu *Wire Sheet* softwaru COACH^{AX}.

Základem této logické vazby je zcela určitě vstupní vyhodnocení provozu (Obrázek 4.2), takto zavedená vazba porovnává dva vstupní signály – *Pracovní den* a *Den 6:00-21:00*. Při splněné podmínce sobě rovnající je na provoz přivedena hodnota *true* nebo *false*, tedy zdali je provoz nebo není.

Na listu *Wire Sheet* jsou nedílnou součástí také pomocné proměnné teploty, do těchto proměnných jsou generovány data za pomocí sinusového signálu nebo nastaveny přímo z prvku *Numeric Schedule*.

Dále po vyhodnocení provozu a vygenerování dat teplot se zhotovil na listu *Wire Sheet* celý chod jednotky VZT. Takto zhotovená logika chodu, je-li VZT v režimu „zapnuto“, porovnává na vstupu teplotu, kterou požaduje uživatel s teplotou v místnosti a výsledek posílá na vyhodnocení chlazení nebo topení, které se bude spouštět.

Dalšími vytvořenými podsložkami jsou složky *alarmy* a *teplota místnosti*. Složka *alarmy* sdružuje poruchy, které mohou za provozu vzniknout, jsou členěny na kritické a nekritické poruchy. Do nekritických poruch spadají např. zanesené filtry, které nevedou k úplnému odstavení VZT, naopak při kritické poruše by operátor musel jednotku VZT úplně odstavit, než by se porucha opravila. Poruchy jsou dále více řešeny také ve složce *simulace dat* na listu *Wire Sheet*, ale to jen za pomocných proměnných, které se přepisují v reakci na uživatelem zadané hodnotě.

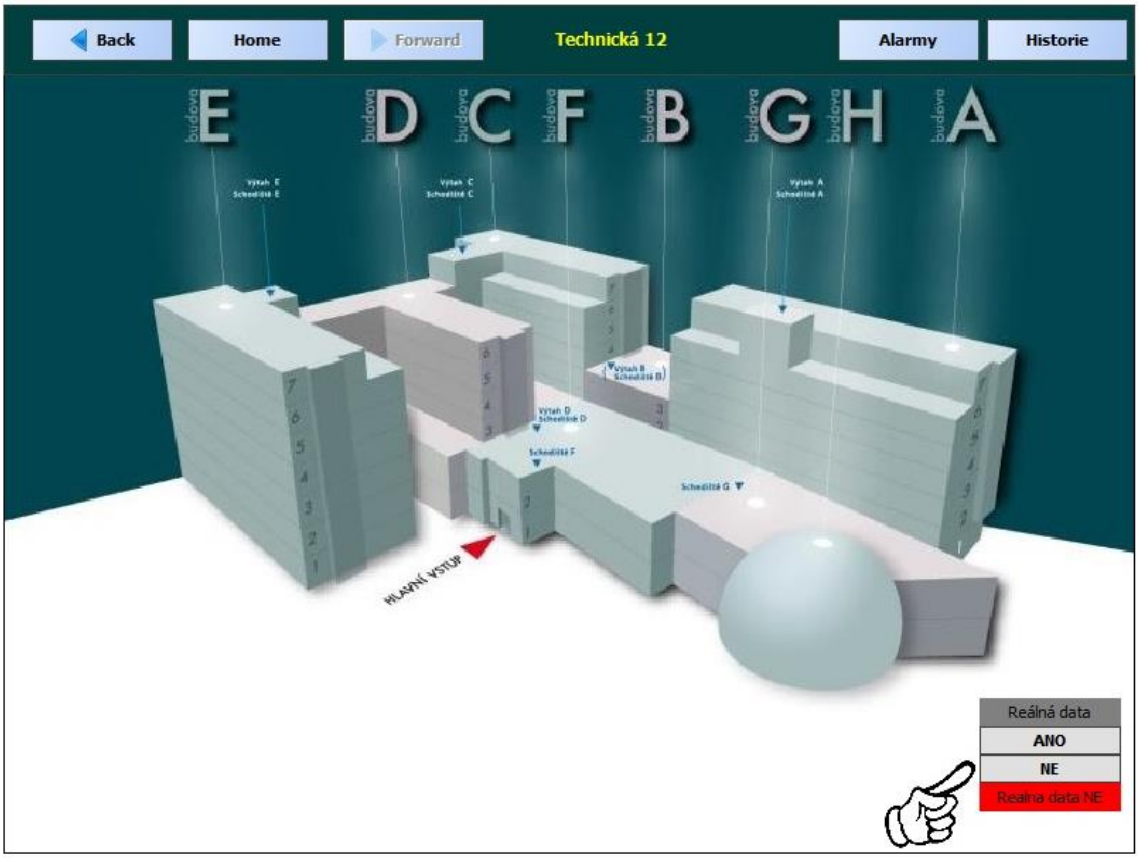
Složka *teplota místnosti* simuluje teplotu místnosti, za pomocí generování hodnot sinusového průběhu, a je vyhodnocena jako průměr čtyř hodnot z čidel v místnosti.

Celou logiku generování dat můžete vidět v Příloze 2.

4.2 **Vizualizace**

Při spuštění vizualizace (Obrázek 4.2) kliknutím v softwaru COACH^{AX} na složku *T12*, můžeme vidět na úvodní obrazovce plánek areálu T12, kde je možné vybrat po kliknutí určitý blok, v našem případě blok H. Úvodní obrazovka obsahuje také menu, které je typické pro všechny následující zobrazená okna. Toto menu obsahuje navigační prvky a položky *Alarmy* a *Historie*. V neposlední řadě je na úvodní obrazovce připraveno tlačítko pro přepnutí na reálná data.

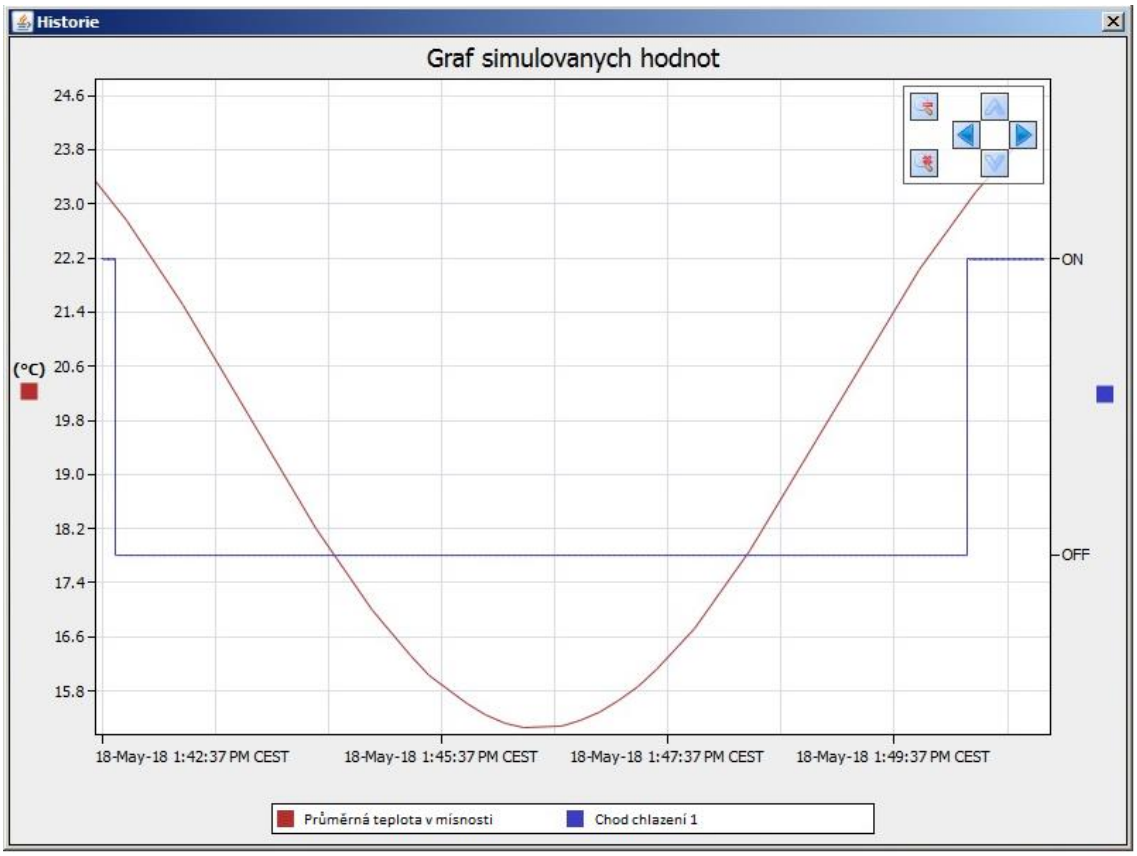
Při vybrání bloku H z úvodní obrazovky se dostaneme na obrazovku pro daný blok. Zde máme k vidění prvky: přehled – teplota místnosti, AHU – zobrazení jednotky VZT, alarmy a v neposlední řadě simulace, která souvisí s generováním dat. Všechny tyto prvky odkazují do dalšího okna vizualizace.



Obrázek 4.2: Úvodní obrazovka vizualizace

Rozkliknutím prvku teplota místnosti se nám rozbalí okno, ve kterém můžeme vidět teploty čidel v jednotlivých rozích místnosti. Prvkem AHU přejdeme do okna zobrazující jednotku VZT a její části v podobě klapek, ventilátorů, chladiče atd. Tyto části jsou tvořeny za pomocí animace, např. při provozu můžeme vidět točící se ventilátor. Dále prvkem alarmy v bloku H otevřeme okno týkajících se částí VZT, které mají zelený indikátor v případě správného chodu nebo červeného. V poslední řadě nám zbývá prvek simulace, který souvisí s generovanými daty. Při rozkliknutí přejdeme do okna „Simulovaná data VZT“, kde díky interaktivnímu chování můžeme ovlivnit generování dat. V tomto okně můžeme např. zadat automatické generování teplot nebo ručně zadat hodnotu teploty, dále můžeme simulovat poruchu a tím zapříčinit alarm, který na sebe upozorní blikající výstrahou v menu. V menu si můžeme následně zobrazený alarm zobrazit a kvitovat jej.

Na závěr bych se ještě vrátil k položce *Historie* ve společném menu. Přes tuto položku se dostaneme k výběru jednotlivých dat MaR budovy T12. Zde si můžeme jednotlivá data vybrat a následně zobrazit jejich průběh v časovém úseku. Ukázku takového zobrazení můžeme vidět viz (Obrázek 4.3).



Obrázek 4.3: Průběh simulovaných dat – Historie

5. ZÁVĚR

Cílem mé práce bylo nastudování sbírání dat z inteligentních budov a s těmito daty pracovat. Úvodem práce byly přiblíženy pojmy inteligentní administrativní budova, používaný protokol BACnet a sbíraná data. Dále jsem se seznámil se softwarem COACH^{AX}, který je nedílnou součástí praktické části práce. V praktická části se věnuji vývoji a následné realizaci databázového modelu MaR dat budovy T12. Pro vývoj byl použit pomocný nástroj CASE studio 2, kde byl navrhnut model databáze a následně přenesen do softwaru MS-SQL Server. Vzniklá databáze byla propojena se softwarem COACH^{AX} a vytvořena logika obsluhy databáze spočívající v použití skript s SQL příkazy. V poslední řadě byla vytvořena simulace generování dat a vizualizace v softwaru COACH^{AX}.

Práci bych zhodnotil jako středně náročnou z hlediska nepříliš dostupných materiálů pro dostatečné pochopení softwaru COACH^{AX}, kdy mi nejvíce času zabrala orientace v softwaru a hledání správného řešení pro danou problematiku.

Vzhledem k tomu, že je software COACH^{AX} psán v jazyce JAVA viděl bych v navazující práci možnosti v podobě doprogramování prvků pro lepší obsluhu databáze a s ní spojenou vizualizaci.

Literatura

[1] VALEŠ, M., Inteligentní dům, 2. vyd., ERA, 2008, ISBN: 978-80-7366-137-3

[2] Automatizace budov – význam a funkce. Buildsys [online]. [cit. 2017-10-29]. Dostupné z: http://www.buildsys.cz/buildsys-systemy-pro-rizeni-budov-automatizace-budov.html

[3] BACnetTM – A standard communication infrastructure for intelligent buildings [online]. In:. [cit. 2018-01-02]. Dostupné z: http://www.bacnet.org/Bibliography/AIC-97/AIC1997.htm

[4] Úvod do BACnetu - Building Automation and Controls Network [online]. [cit. 2017-12-09]. Dostupné z: https://automatizace.hw.cz/uvod-do-bacnetu-building-automation-and-controls-network

[5] Relační databáze. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2018-01-02]. Dostupné z: https://cs.wikipedia.org/wiki/Rela%C4%8Dn%C3%AD_datab%C3%A1ze

[6] MATZ, V., Systémy používané v "inteligentních" budovách: přehled komunikačních protokolů, TZB, [online] [cit. 2017-10-30] Dostupné z: http://vytapeni.tzb-info.cz/mereni-a-regulace/6879-systemy-pouzivane-v-inteligentnich-budovach-prehled-komunikacnich-protokolu

[7] PIVOŇKOVÁ, Alena. Optimalizační algoritmy řídících systému inteligentních budov. Praha, 2005. 78 s. ČVUT. Fakulta elektrotechnická. Vedoucí diplomové práce Ing. Jaroslavu Honců. Dostupný z: https://support.dce.felk.cvut.cz/mediawiki/images/4/47/Dp_2005_pivonkova_alena.pdf

[8] SBĚR DAT PRO VYHODNOCOVÁNÍ SPOTŘEB ENERGIE KLIMATIZAČNÍCH JEDNOTEK [online]. [cit. 2017-12-09]. Dostupné z: https://www.asb-portal.cz/tzb/vetrani-a-klimatizace/sber-dat-pro-vyhodnocovani-spotreb-energie-klimatizacnich-jednotek

[9] Základy relačních databází, jejich využití v programování webu [online]. [cit. 2017-12-30]. Dostupné z: http://gml.vse.cz/data/oppa-webdesign/zaklady-db.html

[10] HONEYWELL. Cap0_IntroduceAX. Tréninkový materiál.

[11] HONEYWELL. CentraLineAX. Tréninkový materiál.

[12] KVASNIČKA, P. Aplikace pro regulátor HAWK. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. 80s. Vedoucí bakalářské práce Ing. Miroslav Jirgl.

[13] KLÍMA, J. Aplikace s regulátorem HAWK. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 65s. Vedoucí práce: Ing. Miroslav Jirgl, Ph.D.

[14] Relační databáze. Materiály pro studenty IVT [online]. [cit. 2018-01-02]. Dostupné z: http://vyuka.greendot.cz/materialy/material-4.pdf

[15] MS-SQL krok za krokem. Itnetwork [online]. [cit. 2018-04-05]. Dostupné z: https://www.itnetwork.cz/ms-sql/mssql-tutorial-uvod-a-priprava-prostredi
[16] Základy SQL [online]. In: . [cit. 2018-10-05]. Dostupné z: http://books.fs.vsb.cz/SQLReference/Sadovski/SQL-PRVN.HTM
[17] Jazyk SQL. In: VOHO = VOJTA HORDĚJČUK [web]. [cit. 2018-10-05]. Dostupné z: http://voho.eu/wiki/sql/
[18] PODOLSKÝ, O. Sběr MaR dat. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 44s. Vedoucí práce: Ing. Radek Štohl, Ph.D.

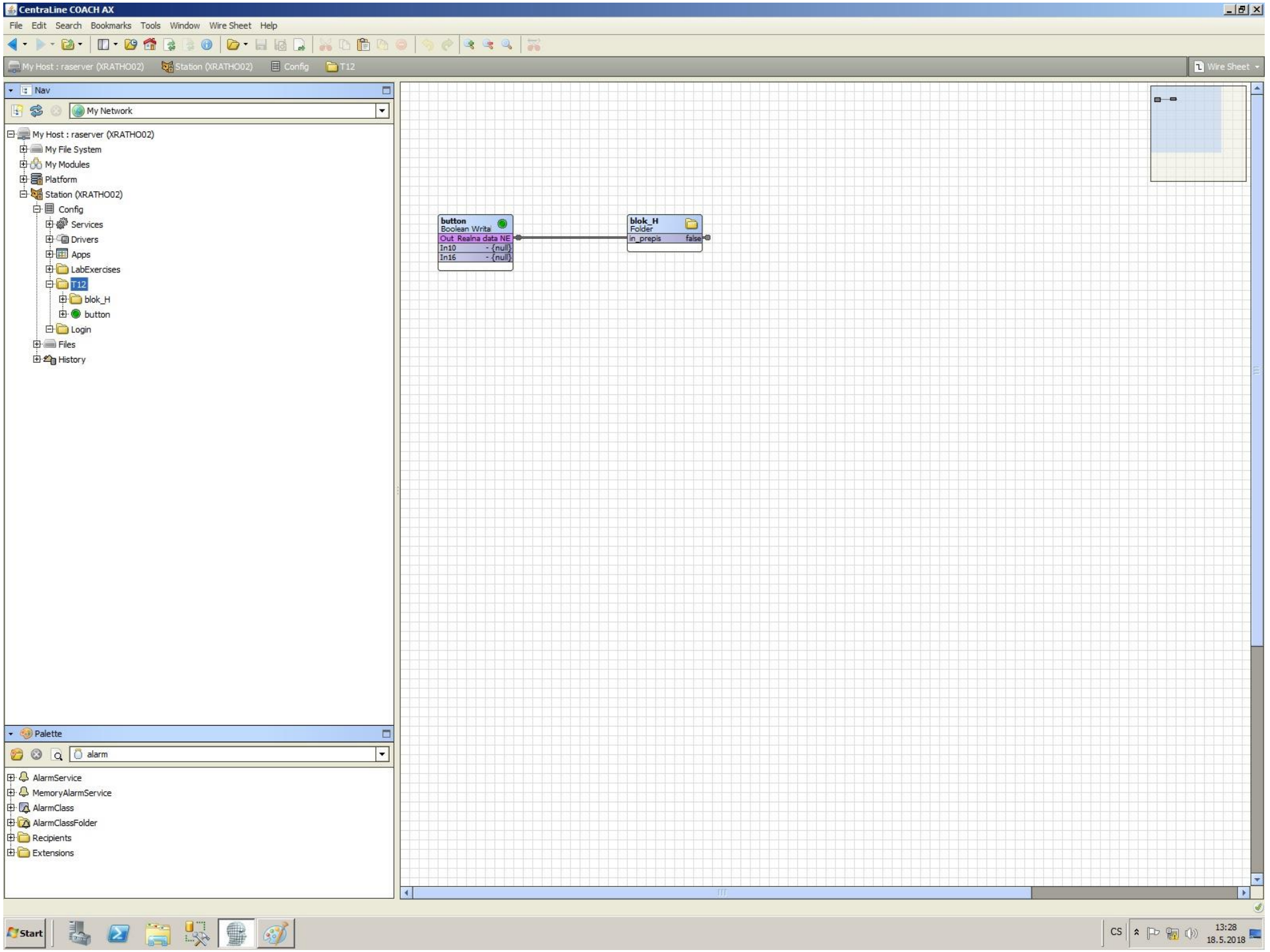
Seznam příloh

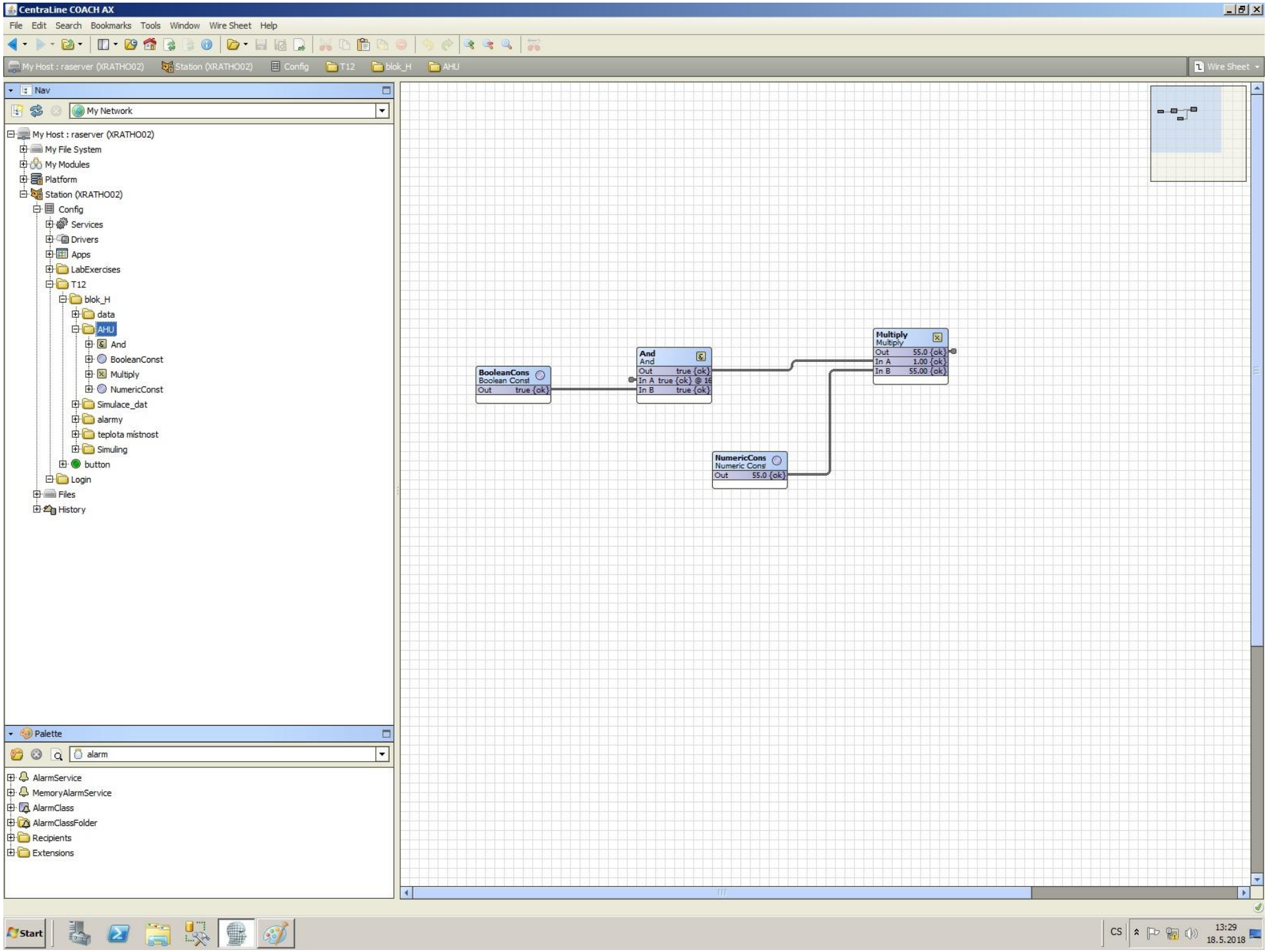
Příloha 1 - Obsah CD

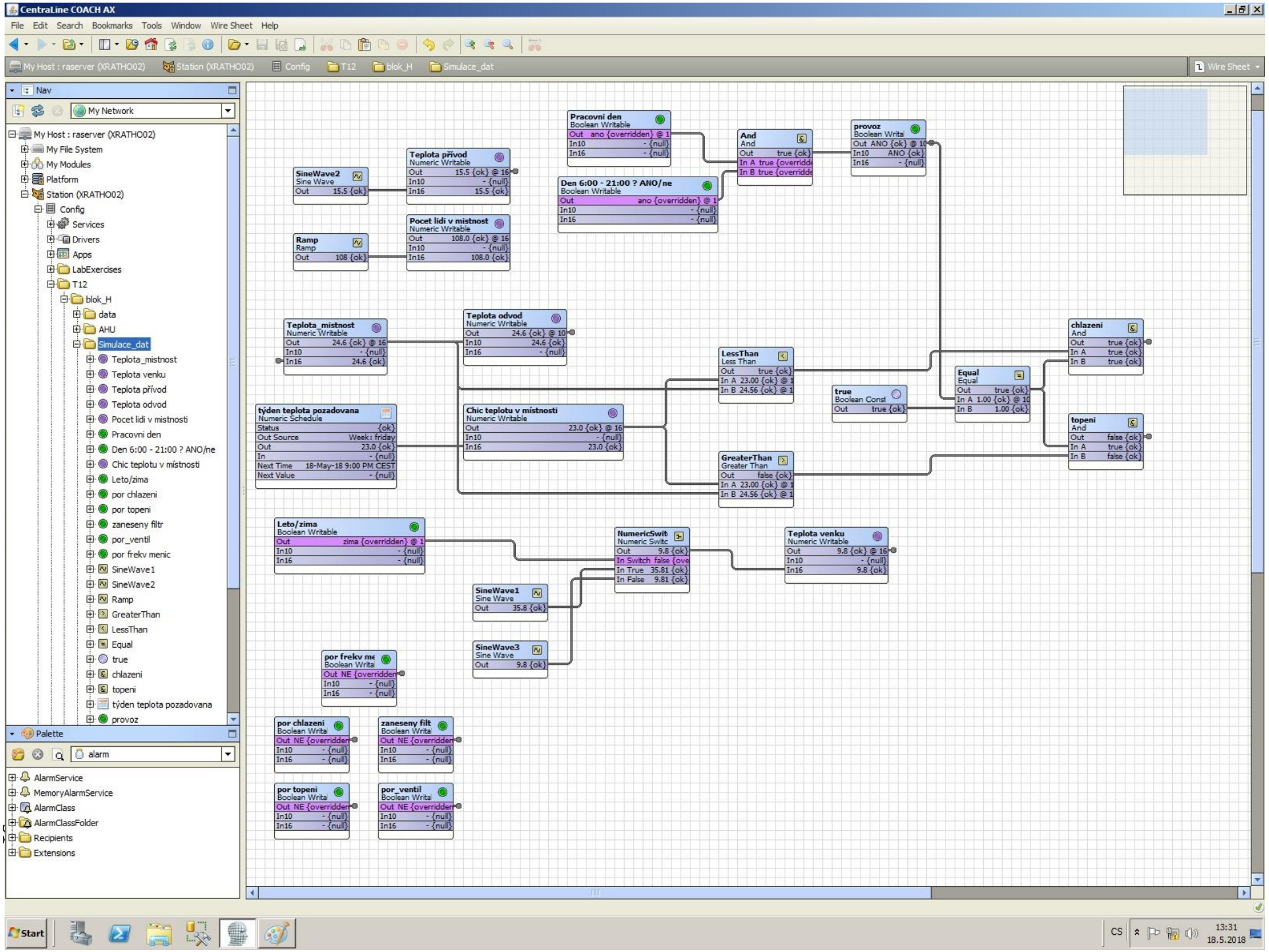
K bakalářské práci je přiloženo CD s touto adresářovou strukturou:

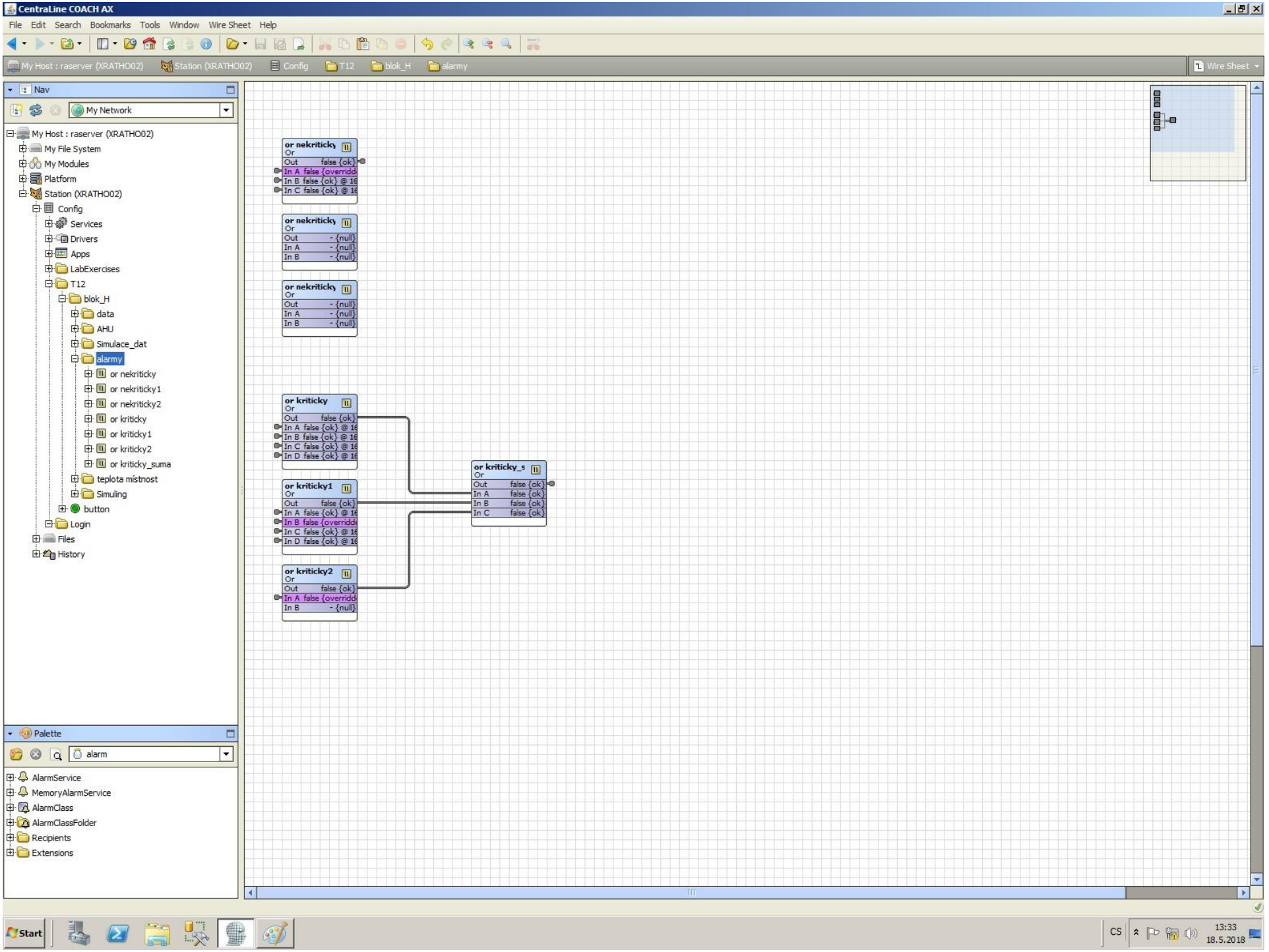
- COACH_STANICE_ZALOHA / - Složka obsahující zálohu stanice.
- T12_MODEL_SKRIPTY / - Složka obsahující skripty pro obsluhu databáze.
- T12_MODEL_MS-SQL / - Složka osahující zálohu databáze.
- soubor T12_MODEL_NAVRH – Navrhnutý model pro sběr MaR dat.
- soubor T12_H_DATA – Tabulka dat bloku H budovy T12.
- soubor BP_MATOUS_RATHOUZSKY_2018 – Tento dokument.

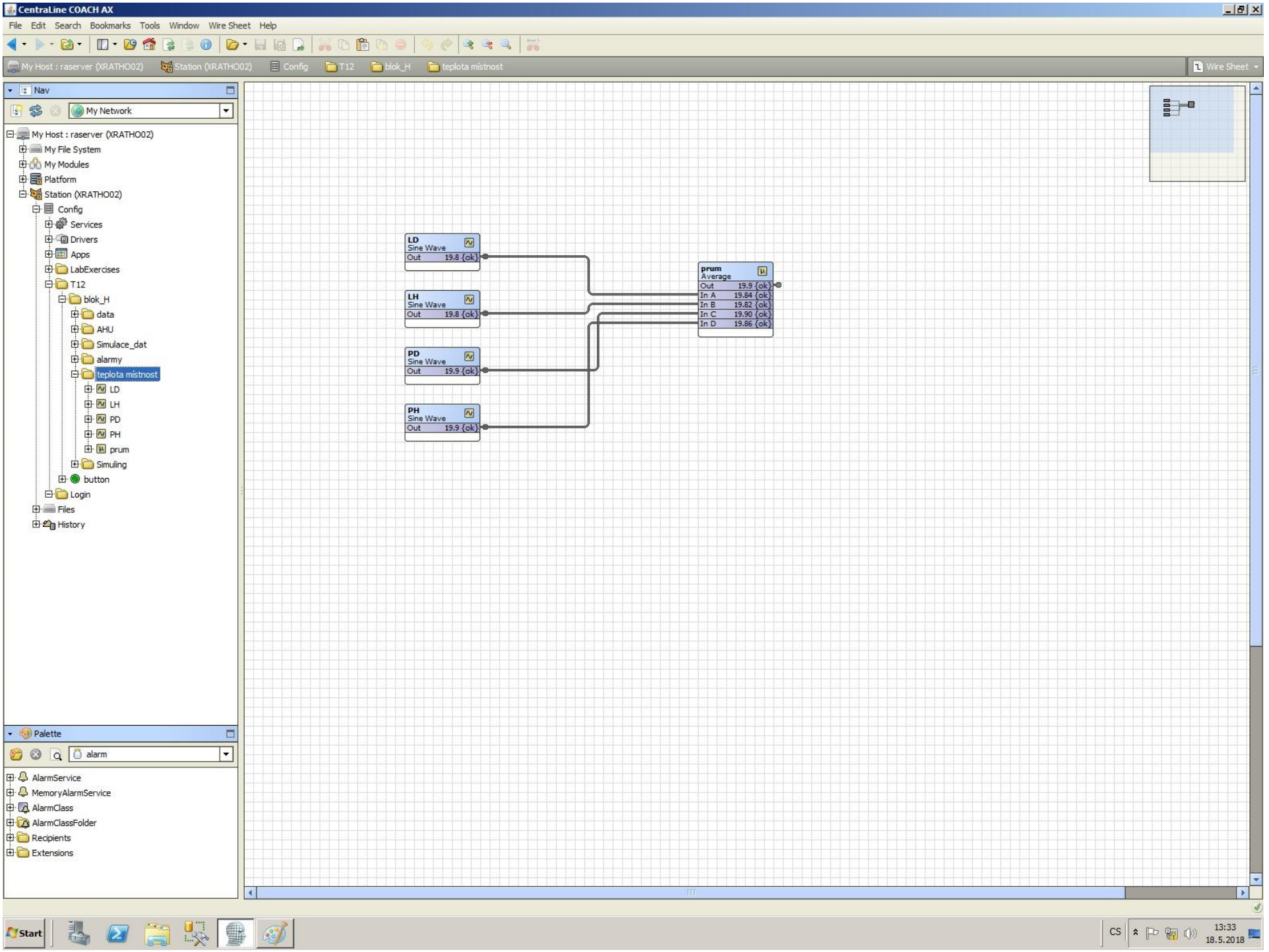
Příloha 2 - Logika generování dat – print screens


CentraLine COACH AX
File Edit Search Bookmarks Tools Window Wire Sheet Help
My Host : raserver (XRATHO02)
Station (XRATHO02)
Config
T12
Wire Sheet
Nav
My Network
My Host : raserver (XRATHO02)
My File System
My Modules
Platform
Station (XRATHO02)
Config
Services
Drivers
Apps
LabExercises
T12
blok_H
button
Login
Files
History
button
Boolean Writa
Out Realna data NE
In10 - {null}
In16 - {null}
blok_H
Folder
in_prepis false
Palette
alarm
AlarmService
MemoryAlarmService
AlarmClass
AlarmClassFolder
Recipients
Extensions
Start
CS
13:28
18.5.2018


CentraLine COACH AX
File Edit Search Bookmarks Tools Window Wire Sheet Help
My Host : raserver (XRATHO02)
Station (XRATHO02)
Config
T12
blok_H
AHU
Wire Sheet
Nav
My Network
My Host : raserver (XRATHO02)
My File System
My Modules
Platform
Station (XRATHO02)
Config
Services
Drivers
Apps
LabExercises
T12
blok_H
data
AHU
And
BooleanConst
Multiply
NumericConst
Simulace_dat
alarmy
teplota místnost
Simuling
button
Login
Files
History
Palette
alarm
AlarmService
MemoryAlarmService
AlarmClass
AlarmClassFolder
Recipients
Extensions
BooleanCons
Boolean Const
Out true {ok}
And
And
Out true {ok}
In A true {ok} @ 16
In B true {ok}
Multiply
Multiply
Out 55.0 {ok}
In A 1.00 {ok}
In B 55.00 {ok}
NumericCons
Numeric Cons
Out 55.0 {ok}
Start
CS
13:29
18.5.2018


CentraLine COACH AX
File Edit Search Bookmarks Tools Window Wire Sheet Help
My Host : raserver (XRATHO02)
Station (XRATHO02)
Config
T12
blok_H
Simulace_dat
Wire Sheet
Nav
My Network
My Host : raserver (XRATHO02)
My File System
My Modules
Platform
Station (XRATHO02)
Config
Services
Drivers
Apps
LabExercises
T12
blok_H
data
AHU
Simulace_dat
Teplota_mistnost
Teplota venku
Teplota přívod
Teplota odvod
Pocet lidi v mistnosti
Pracovni den
Den 6:00 - 21:00 ? ANO/ne
Chic teplotu v místnosti
Leto/zima
por chlazeni
por topeni
zaneseny filtr
por_ventil
por frekv menic
SineWave1
SineWave2
Ramp
GreaterThan
LessThan
Equal
true
chlazeni
topeni
týden teplota pozadovana
provoz
Palette
alarm
AlarmService
MemoryAlarmService
AlarmClass
AlarmClassFolder
Recipients
Extensions
Pracovni den
Boolean Writable
Out ano {overridden} @ 1
In10 - {null}
In16 - {null}
And
And
Out true {ok}
In A true {overridde
In B true {overridde
provoz
Boolean Writa
Out ANO {ok} @ 10
In10 ANO {ok}
In16 - {null}
Teplota přívod
Numeric Writable
Out 15.5 {ok} @ 16
In10 - {null}
In16 15.5 {ok}
SineWave2
Sine Wave
Out 15.5 {ok}
Den 6:00 - 21:00 ? ANO/ne
Boolean Writable
Out ano {overridden} @ 1
In10 - {null}
In16 - {null}
Pocet lidi v mistnost
Numeric Writable
Out 108.0 {ok} @ 16
In10 - {null}
In16 108.0 {ok}
Ramp
Ramp
Out 108 {ok}
Teplota_mistnost
Numeric Writable
Out 24.6 {ok} @ 16
In10 - {null}
In16 24.6 {ok}
Teplota odvod
Numeric Writable
Out 24.6 {ok} @ 10
In10 24.6 {ok}
In16 - {null}
LessThan
Less Than
Out true {ok}
In A 23.00 {ok} @ 1
In B 24.56 {ok} @ 1
chlazeni
And
Out true {ok}
In A true {ok}
In B true {ok}
true
Boolean Const
Out true {ok}
Equal
Equal
Out true {ok}
In A 1.00 {ok} @ 10
In B 1.00 {ok}
topeni
And
Out false {ok}
In A true {ok}
In B false {ok}
týden teplota pozadovana
Numeric Schedule
Status {ok}
Out Source Week: friday
Out 23.0 {ok}
In - {null}
Next Time 18-May-18 9:00 PM CEST
Next Value - {null}
Chic teplotu v místnosti
Numeric Writable
Out 23.0 {ok} @ 16
In10 - {null}
In16 23.0 {ok}
GreaterThan
Greater Than
Out false {ok}
In A 23.00 {ok} @ 1
In B 24.56 {ok} @ 1
Leto/zima
Boolean Writable
Out zima {overridden} @ 1
In10 - {null}
In16 - {null}
NumericSwit
Numeric Switc
Out 9.8 {ok}
In Switch false {ove
In True 35.81 {ok}
In False 9.81 {ok}
Teplota venku
Numeric Writable
Out 9.8 {ok} @ 16
In10 - {null}
In16 9.8 {ok}
SineWave1
Sine Wave
Out 35.8 {ok}
SineWave3
Sine Wave
Out 9.8 {ok}
por frekv me
Boolean Writa
Out NE {overridden
In10 - {null}
In16 - {null}
por chlazeni
Boolean Writa
Out NE {overridden
In10 - {null}
In16 - {null}
zaneseny filt
Boolean Writa
Out NE {overridden
In10 - {null}
In16 - {null}
por topeni
Boolean Writa
Out NE {overridden
In10 - {null}
In16 - {null}
por_ventil
Boolean Writa
Out NE {overridden
In10 - {null}
In16 - {null}
Start
CS
13:31
18.5.2018


CentraLine COACH AX
File Edit Search Bookmarks Tools Window Wire Sheet Help
My Host : raserver (XRATHO02)
Station (XRATHO02)
Config
T12
blok_H
alarmy
Wire Sheet
Nav
My Network
My Host : raserver (XRATHO02)
My File System
My Modules
Platform
Station (XRATHO02)
Config
Services
Drivers
Apps
LabExercises
T12
blok_H
data
AHU
Simulace_dat
alarmy
or nekriticky
or nekriticky1
or nekriticky2
or kriticky
or kriticky1
or kriticky2
or kriticky_suma
teplota místnost
Simuling
button
Login
Files
History
Palette
alarm
AlarmService
MemoryAlarmService
AlarmClass
AlarmClassFolder
Recipients
Extensions
or nekriticky
Or
Out false {ok}
In A false {overridd
In B false {ok} @ 16
In C false {ok} @ 16
or nekriticky
Or
Out - {null}
In A - {null}
In B - {null}
or nekriticky
Or
Out - {null}
In A - {null}
In B - {null}
or kriticky
Or
Out false {ok}
In A false {ok} @ 16
In B false {ok} @ 16
In C false {ok} @ 16
In D false {ok} @ 16
or kriticky1
Or
Out false {ok}
In A false {ok} @ 16
In B false {overridd
In C false {ok} @ 16
In D false {ok} @ 16
or kriticky2
Or
Out false {ok}
In A false {overridd
In B - {null}
or kriticky_s
Or
Out false {ok}
In A false {ok}
In B false {ok}
In C false {ok}
Start
CS
13:33
18.5.2018


CentraLine COACH AX
File Edit Search Bookmarks Tools Window Wire Sheet Help
My Host : raserver (XRATHO02)
Station (XRATHO02)
Config
T12
blok_H
teplota místnost
Wire Sheet
Nav
My Network
My Host : raserver (XRATHO02)
My File System
My Modules
Platform
Station (XRATHO02)
Config
Services
Drivers
Apps
LabExercises
T12
blok_H
data
AHU
Simulace_dat
alarmy
teplota místnost
LD
LH
PD
PH
prum
Simuling
button
Login
Files
History
LD
Sine Wave
Out 19.8 {ok}
LH
Sine Wave
Out 19.8 {ok}
PD
Sine Wave
Out 19.9 {ok}
PH
Sine Wave
Out 19.9 {ok}
prum
Average
Out 19.9 {ok}
In A 19.84 {ok}
In B 19.82 {ok}
In C 19.90 {ok}
In D 19.86 {ok}
Palette
alarm
AlarmService
MemoryAlarmService
AlarmClass
AlarmClassFolder
Recipients
Extensions
Start
CS
13:33
18.5.2018